

Computing *with the* **AMSTRAD**

No. 1
January 1985
£1

A Database Publication

The independent magazine for CPC 464 users

Games galore!

Save Smiley from the Grumpies
Help trap the maze monsters
Crack the secret code
...and more!

**Have fun
with the
Amstrad's
superb
colour
and sound**

**14
software
releases
reviewed**

Amstrad teach-in

Baffled by Basic...mystified by machine code? We show how easy they really are!



ANIROG

*The Name
For Quality
And
Innovation*

Flight Path 737



ADVANCED PILOT TRAINER

NOW AVAILABLE For The **AMSTRAD** £6.95

TRADE ENQUIRIES: ANIROG SOFTWARE LTD. 29 WEST HILL DARTFORD KENT (0322) 92513/8
MAIL ORDER: 8 HIGH STREET HORLEY SURREY 24 HOUR CREDIT CARD SALES HORLEY (03604) 6083
PAYMENT BY CHEQUE P.O. ACCESS/VISA 50p POSTAGE & PACKAGING

SUPERB SOFTWARE FOR THE

AMSTRAD

SUPERB SOFTWARE FOR THE

electron
B.B.C. MICRO

GHOULS

Run through the creepy mansion dodging ghostly ghouls and bouncing spiders. Leap over poison-smearing spikes, scamper along moving platforms and contracting floorboards, and use powerful springs to propel you onto overhanging ledges. Four screens.



Amstrad and Commodore versions £6.95
BBC and Electron versions £7.95
BBC and Commodore Disk price £9.95

Amstrad version

WATCH OUT
FOR OUR NEW
PACKAGING AND
CATALOGUE

MICRO
POWER

MICRO POWER LTD.
SOFTWARES DIVISION, PO BOX 100
LONDON W1A 0AA. TEL: 01-253 4000
TELETYPE: 01-253 4000. FAX: 01-253 4000
© 1988 MICRO POWER LTD.





Lower prices!

Now better value for money
than any other computer magazine
in the UK.

Best for the
AMSTRAD
owner
and user

76
pages
of
advice
and
information

AMSTRAD/POWER & ADVICE is now... **COMPUTING WITH THE AMSTRAD**

Vol. 1 No. 1 January 1985

Managing Editor: **Barak Meakin**

Features Editor: **Peter Biley**

The A Team:

Mike Biley

Alan McLehden

Kevin Edwards

Production Editor: **Peter Glover**

Layout Design: **Heather Sheldrick**

Reviews Editor: **Mike Cowley**

Advertisement Manager: **John Helling**

Advertising Sales: **Margaret Clarke**

Editor in Chief: **Peter Brunsell**

Editorial: 061-486 6836

Administration: 061-486 6363

Advertising: 061-486 6500

Subscriptions: 061-486 6171

Telex: 887664 SHARST G

Postal Mailbox: 854500383

Published by:

Databases Publications Ltd,
Europe House, 58 Chester Road,
Hazel Grove, Stockport SK7 5NY.

Subscription rates for
52 issues, post free:

£12 - UK

£15 - Eire (airmail only)

£20 - Rest of world (airpost)

£40 - Rest of world (airmail)



Member of Audit
Bureau of Circulations

"Computing with the Amstrad" welcomes program listings and articles for publication. Material should be typed or computer-printed, and preferably double-spaced. Program listings should be accompanied by source tape or disc. Please enclose a stamped, self-addressed envelope, otherwise the return of material cannot be guaranteed. Considerations requested for publication by Database Publications Ltd will be on an all-rights basis.

© 1985 Database Publications Ltd. No material may be reproduced in whole or in part without written permission. While every care is taken, the publishers cannot be held legally responsible for any errors in articles, listings or advertisements.

"Computing with the Amstrad" is an independent publication and neither Amstrad Consumer Electronics plc or Amstrad are responsible for any of the articles in this issue or for any of the opinions expressed.

News trade distribution:

Successes Sales and Distribution Limited, 11 Brighton Road, Exeter, Devon EX4 6AP. Tel: 0323 527655.

7 NEWS

Keep up to date with the latest happenings in the busy, expanding world of the Amstrad computer.

9 AMSTRAD ANALYSIS

Join Trevor Roberts as he casts his eagle eye over some short, simple programs. You're bound to find some useful ideas here.

10 FIRST STEPS

Programming is nowhere near as hard as the experts like to make out. This easy-to-follow article shows you what we mean.

15 GRAPHICS

Make the most of the Amstrad's amazing colour when they're explained as well as this...



20 SOUND SENSE

We show you how to become a big noise in the far from silent world of the Amstrad music.

23 LOOPHOLES

Having problems typing in listings? You'll find this light-hearted advice gives you all the help you'll need.

26 TRAPPER

Can you trap the mouse monster in this fast action game? It's not as easy as it seems. And if you can tear yourself away from the action there are a lot of programming tricks to be learned from the listing.



28 GRAPHICS REFERENCE

Puzzled by MODE, INK, and PEN? The first of our ready reference guides will help make life easier.

29 SOFTWARE SURVEY

The latest software releases for the CPC464 are assessed by our team of frank and thorough reviewers.

34 SMILEY v GRUMPIES

Can you avoid the Grumpies as you guide Smiley round the labyrinth? We guarantee hours of fun from this reincarnation of the arcade classic.



38 BITS AND BYTES

We lift the lid on binary in the first of our regular looks at the way the CPC464 does its sums.

41 LETTER LITTER

Collect the rubbish – and learn your way around the keyboard – in this entertaining educational game for all the family.

45 SECRET CODE

Can you crack our secret code and turn the Grumpies into Smilies? Yes, it's an old favourite brought up to date for the Amstrad!

46 AMSTRAD EXPANSION

Make the most of your mighty micro by giving it a bit more muscle. We investigate the ins and outs of the CPC464.

50 AL'S BEAT

Our tame Mr Piod pounds his regular Amstrad beat and what does he come up with? A program that prints "Hello, hello, hello ..."

54 BOOK REVIEW

We take an in-depth look at one of the latest books aimed at the beginner to Amstrad computing.

56 MACHINE CODE

Don't tell the know-alls, but machine code is really quite simple – that is, when it's explained as clearly as this.



61 SCROLLER

Add that professional touch to your text with this slick sideways scrolling routine.

65 SUBSCRIPTIONS

No false modesty – you've seen how good our mag is so why leave getting it to chance when you can subscribe? And there's a special introductory offer!

65 POSTBAG

Believe it or not, the letters were flooding in even before we went to press. Here are just a few of the liveliest.

Introducing ourselves...

WELCOME to the first issue of *Computing with the Amstrad* – the new monthly magazine written by users of the CPC464 for users of the CPC464.

We're big, bright and completely independent, so what you read here is totally impartial. For instance, our program reviews tell it like it is – not how the software houses would like it to be.

Actually, we're quite impressed with the standard of software available for the CPC464 as you'll see from our review pages.

Our one quibble is that, good though the programs are, too many show signs of having been quickly translated from other computers. All too often they fail to take full advantage of the Amstrad's unique features and potential.

But you'll find much more than reviews inside this issue. As you'll see from this page, the magazine's packed with original programs and informative articles.

We know that many of you are newcomers to micro-

computing, so we've got plenty to interest the novice – including easy-to-follow introductory articles on the Amstrad's Basic, Search and Graphics.

And we've not forgotten those of you with more experience. You'll find plenty to interest you, including an introduction to machine code and a survey of the Amstrad's hardware, as well as lots of useful ideas and routines you can incorporate into your own programs.

One thing we're sure of is that you've got your own ideas of what the magazine should cover. We're always open to suggestions – so let's hear yours.

And we're on the lookout for writers, too. If you've got a program you think would interest us let us know. You won't make a fortune, but you'll have a lot of fun.

And thanks for all the letters – keep them coming.

See you next month.

The A Team

Amstrad Adventures from Interceptor Software.

R.R.P.
£6.00
EACH

Interceptor Software are proud to present their first 4 releases on the Amstrad. These 'user friendly' graphical adventures not only utilise the memory to a full but display the best graphics yet seen on this new computer. A sensible price of £6.00 each makes them the best value software on the Amstrad.



Forest of Worlds End

Forest of Worlds End is a mythical adventure where you have to rescue Princess Maria who has been captured by the evil wizard, Ecto. Many have sought you in the forest!



Heroes of Karn

Heroes of Karn is the 1st adventure on the CBM 64 has ever been converted to the Amstrad. The classical story will keep you occupied for many hours and when you finally solve the adventure and find the Heroes you will find a magnificent final life story to be filled with the follow on, Empire of Karn which will be released in 1985.



Jewels of Babylon

Jewels of Babylon is set with a great theme, where you, the sole survivor of a pirate raid have the task of recovering the Jewels for Queen Victoria who has promised them as a wedding gift to an Italian Prince.



Message From Andromeda

Message From Andromeda is a space adventure where you can a captain of a pirate vessel in the galaxy. The task within you is that thought with danger! Be prepared for the unexpected.

Available from all leading Software Retailers or direct from

**INTERCEPTOR
SOFTWARE**

Linton House, The Green, Tadley, Hants, England.

Tel: (07156) 71143/3711 Telex: 849101 INMICS G



Stand back, sliced bread

IS Amstrad the greatest thing since sliced bread? Inside the company they've no doubts.

Company chairman Alan Sugar says: "In all the years I have been part of this industry I have never experienced such a demand for any one product."

"The enquiries from home

and overseas have been quite overwhelming and supply is nowhere near demand".

Ray Cope, head of the computer division at distributors Europa Electronics, says: "I have never in all my experience in distribution and sales seen a product take off like this".



Alan Sugar: "Never experienced such a demand".

Sophisticated new projects on way

BEHIND the CPC464 stands a highly profitable international organisation. Group turnover of Amstrad Consumer Electronics in 1983/84 was £64.9 million - up more than £30 million from the

previous year.

Group pre-tax profit was £16.1 million compared with £8 million, and exports more than doubled from £5.3 million to £10.8 million.

These figures cover sales of the company's packed audio-cassets, colour TVs and video recorders and were compiled before the CPC464 appeared in the shops.

Exciting

Projected computer sales are 200,000 in 1984 and 400,000 worldwide in 1985 - together with peripheral hardware to complement the machine.

In his annual report, Amstrad chairman Alan Sugar promised: "There are exciting plans under development in this sector for new items to be launched in the third quarter of 1985."

"These new projects will incorporate more sophisticated electronic technology and keep Amstrad ahead of the market".

Did you know?

AMSTRAD gets its name from the abbreviation of Alan M. Sugar Trading, which started manufacturing electronic equipment in 1968. Alan Sugar is the present chairman and chief shareholder of Amstrad.

Campaign postponed

THE CPC464 has made a huge impact on the computer marketplace with very little advertising backup.

Demand for the machine has been so great that a planned £2 million nationwide TV campaign has been postponed until early 1985.

Instead, Amstrad has been restricting its advertising to national daily papers and specialist magazines.

The reason, says a spokesman, is "the Amstrad reputation for value for money products is second to none."

"Many faithful customers owning other Amstrad products are now buying the computer."

"Since we started deliveries of the first computers in June, the product has made a wonderful reputation with both trade and user."

"To have continued with our TV advertising in the autumn would have complicated the supply position."

"But by a big blast starting in the New Year we can be assured of a good spring sales level".

Wally moves in

WALLY Week, star of Micro-Gem's *Automata* and *Pyjamarama*, has set his sights on becoming a favourite of Amstrad users. *Pyjamarama* is now available for the CPC464 at £8.95.



Fairytale adventure

READ the story, play the game - that's the theme of a novel Amstrad package from Database Publications.

The Magic Sword is an adventure game on cassette, specially written for young children. But that's only half the story.

With it comes a colourful 48 page book which tells all the events leading up to the start of the adventure.

All the elements of a traditional fairy story are there - a handsome prince, a beautiful princess, a white castle with secret passages and mysterious dungeons, fierce forests, deep

swims - and a crooked house where lives a wicked witch.

All these are in the game as well, plus colourful animated graphics and lots of thrilling sound effects which encourage the child to travel through the countryside and explore the castle to find the princess and rescue her from the clutches of the witch.

The program is the brain child of Newcastle's Kristin Halls and her son Martin, 13.

Kristin wrote the book with help from Martin, who drew the pictures. Martin devised the game program based on his mother's design and graphics.



Goodies galore lined up

Following a completely new machine, the CPC484 is surprisingly well provided with peripherals and software.

□ Amstrad's own 80 column dot matrix printer, uses a standard parallel centronics interface for text processing.

The GMP1 operates at a print speed of 50 cps and includes instruction extensions that provide dot-addressable graphics and full screen dump capability.

The self-aligning tractor feed is adjustable from 4.5in to a maximum paper width of 10in.

□ A printer lead, price £9.95, is available for use on non-Amstrad printers.

□ Selling at £39.95, the Amstrad printer module also allows the CPC keyboard to be used with a domestic colour TV set.

□ The Amstrad joystick's design allows two to be operated from one port. Cost is £14.95 a unit.

□ The Amstrad disc drive costs £199.99 and comes complete with interface, PSU, CPM and Dr Logo operating systems.

□ Firmware and Basic specification manuals retail at £19.95 each.

Amstrad, the specialist division set up to market the software support has more than 80 arcade style games available now or due for release shortly. Most cost £9.95.

About a dozen educational and tutorial programs are available, ranging in price from £9.95 to £35.

Amstrad drive for schools market

AMSTRAD is going all out to capture a major share of the education market from Acorn and RML.

And if all goes to plan over the next few months the CPC 484 could become the schools' computer of the future.

Amstrad admits it is currently short of software announcements in the educational sector - a situation it aims to quickly rectify - but insists its prices and product are potent weapons.

First move in the campaign was the recent appointment of Northern Computers as exclusive educational distributor for the CPC484.

Northern's education and training division director Gareth Usher has driven up his battle plan.

Primary target is the ageing RML 485C whose capabilities are "virtually matched by the CPC at half the price" according to Usher.

Convert

Northern is asking 500 publishers of RML educational software to convert their programs for the Amstrad. "Not a very difficult task given the similar capabilities of the two machines", says Usher.

His next task is to get 300 BBC software publishers to do likewise. "They will have to re-write the programs, but we have twice the memory to play with", says Usher.

In a third line of attack



Northern Computers' Gareth Usher - getting a foot in the education door

Northern has already approached all Britain's 35,000 schools, colleges and universities - giving them information about the product and Amstrad's plans for its future.

One the main selling point has been price. Schools can buy a green screen CPC484 for £189 or a colour version for £219.

"We can supply a lab with 10 colour Amstrads for what it costs to buy 10 BBC Micros with colour", says Usher.

"We have already placed large numbers of machines in educational establishments, with a particularly big response from polytechnics.

"Another big selling point is our free one year service and

maintenance contract".

Patently in the educational press has been particularly helpful. "The machine has had a wonderful reception from educationalists", says Usher.

Another big boost will come very shortly when Amstrad brings out its interface for boomers, enabling the CPC484 to join BBC Micro networks in schools.

Amstrad is also testing the language and CPM compatibility of its machine.

First drives to be made available are the 3in version with CPM and Logo.

But soon the CPC484 will have Microgate 5in disc drives, also with CPM and Logo.

Rush to board the bandwagon

The runaway success of the CPC484 left many computer dealers biting their nails as the Christmas sales period loomed.

The reason - their applications to climb onto the fast rolling Amstrad bandwagon were piling up at the offices of Europa Electronics, sole distributors to independent retailers.

With the seasonal spending spree about to begin, 450 dealers had received their credit clearance and been registered by Europa.

But there was a backlog of about 100 applications from retailers desperate to stock the fast-selling machines.

Ray Coops, head of the Europa computer division explained: "A

major cause of the delay in processing applications is that so many come from dealers we haven't worked with before and they nominate our competitors as credit references.

"I'm sure our competitors would not deliberately sit on these applications - it's more likely a case of us overloading them with paperwork".

Europa had planned to release 25,000 machines to independent dealers but Amstrad was hoping to boost this total to 40,000.

Added to the stocks supplied to major chains like W.H. Smith, Boots, Harrodians and Comet, it adds up to a projected 1984 sales figure of 200,000 for the CPC484.

ANALYSIS is the first of a regular series where short, simple programs come under scrutiny. This month's program uses the Amstrad's character set to put a dancing man on the screen.

Amstrad Analysis

By Trevor Roberts



Put the little man on the screen

Subroutine

Call the subroutine

Delay loop

Position text cursor

```

10 REM DANCING MAN
20 REM By: STEWART HUGH
30 MODE 1
40 CLS
50 GOSUB 120
60 PRINT CHR$(205)
70 GOSUB 120
80 PRINT CHR$(206)
90 GOSUB 120
100 GOSUB 120
110 GOSUB 120
120 PRINT CHR$(205)
130 GOTO 30
140 REM*****
150 REM position man
160 FOR delay=1 TO 100
170 NEXT delay
180 LOCATE 20,10
190 RETURN
200 REM*****
  
```

10,20

Two REM statements saying what the program is called and who wrote it.

30

Selects Mode 1.

40

CLEAR the screen.

50,70,90,110

Call the subroutine which starts at line 120 and ends at 190.

60,80,100,120

These lines do the work of printing the little man on the screen. The differing figures in the brackets of the CHR\$() produce a different 'little' man.

130

The GOTO sends the program back to line 50, causing the program to repeat endlessly.

140,200

The REMs and asterisks are just used to show where the subroutine is hidden.

150-190

150

Form the subroutine.

Makes a note of what the subroutine does.

160-170

The FOR ... NEXT loop just slows things down a little so we have time to see the man.

180

The LOCATE makes sure that the man is printed at the same position each time the subroutine is called. As the different characters overlap each other, so the man appears to dance.

190

The RETURN ends the subroutine, sending the Amstrad back to the line following the line that called the subroutine.

standard typewriter keyboard surrounded by several additional keys.

However, unlike the standard typewriter, not only do the numbers appear on the top row of keys as usual, but also on a separate "numeric keypad" to the right of the main keyboard.

Notice that the keyboard has separate keys for the letter O and the number 0. The 0, as you'll see if you try it, always appears with a diagonal line across it — whether you type it on the main keyboard or the numeric keypad.

You must keep O and 0 entirely separate. I guarantee that a lot of your early mistakes in programs will be caused by typing O instead of 0!

On the same line, notice that there are special keys for I. Don't use the lower case "I" for one, as you have to on some typewriters — the mice won't appreciate it!

Above the numeric keypad there's a cluster of five keys — one labelled COPY, surrounded by four keys with arrows. These are simply dealt with — we're going to ignore them for the present.

Going back to the main keyboard, you'll see that as well as the letters and numbers ("alphanumeric") as the jargon has it there are other keys, neatly picked out in colour.

Some, such as Caps Lock and Shift, you'll be familiar with from ordinary typewriters. Others, such as Ctrl and Esc, should be new to you.

Let's introduce a convention to make life easier. If I want you to press the Shift key, I'll ask you to type

[Shift]

If, on the other hand, I ask you to type

Shift

I want you to type S followed by i, followed by l and so on. So if I want you to press a single key, I'll put the name of that key in square brackets — otherwise type each character out separately.

Now Enter is quite an important key — it's the blue one at the right of the main keyboard. It's also at the bottom right-hand corner of the numeric keypad.

We use Enter in a similar manner to the Return key on an electric typewriter, to ensure that the typing continues on a new line.

It's far more important than that,

You can't hurt the Amstrad by accidentally mistyping something — so feel free . . .

though, Enter not only gives you a new line, it also sends the message you have typed on that line into the computer to be acted upon.

If you've been following so far, you should have a screen looking something like this:



If, however, you've been idling until now, type a few letters, then press [Enter]. Odds on, you'll get a message back from the computer saying:



Don't worry about this message — you can't hurt the mice by accidentally mistyping something, so feel free to experiment.

All that Syntax error means is that the computer doesn't understand the words you've just sent it. You see, it needs to be talked to in its own language, called Basic.

However learning Basic isn't like learning a genuinely foreign language. Basic is very similar to English, but it only uses a small set of selected English words in order to make things simpler for the computer. These selected words are called keywords.

This, by the way, is why I said it was "odds on" you'd get the Syntax error message when you pressed Enter. You might, by chance have hit

on a Basic keyword.

For example, you can mark the end of a Basic program with the keyword end. The people who designed Basic could have chosen the word Finish to do this.

Let's see if there's any difference in the words. Type end and press [Enter]. Then type Finish and press [Enter]. The screen will look like this:



Notice the difference? The CPC464 accepts end, but not Finish.

Admittedly, end doesn't accomplish very much. After all, you haven't anything in there to end, have you? But at least the mice didn't hurt the message Syntax error at you as it did with Finish.

This is because end is a Basic keyword, while Finish isn't.

Notice something about the Syntax error message — it has a capital S. So far everything we've typed in has been in lower case — at least in theory. (Remember, I said stick to the letters of the alphabet!)

I bet you've already discovered that if you hold Shift down while you're typing the letters appear in upper case. If not, try it now. It's what you'd expect, if you've ever used a typewriter.

Again it will come as no surprise to see that if you press the



key with Shift, you get the "S", while un-Shifted you get the "s".

If you press Caps Lock and release

If you'll find that the letters of the alphabet come out in upper case automatically. Also if a key has two characters (or legends as they are known), you get the upper one on the screen.

So, with the



key, you'll get "G" with Caps Lock on.

This upper case output will continue until you press Caps Lock again, then you'll go back to normal. Press it again and you're back into upper case. Once more and it's normal again. (This way of jumping between one "state" and another is called "toggling" in computer jargon.)

What do you suppose would happen if you pressed END instead of end? Try it and see. One of the nicest things about using a computer is that you don't have to speculate. If you find yourself asking "What would happen if?" or "I wonder if this would work?" you can go right ahead and try it.

As I said, you can't hurt the micro from the keyboard, so go ahead and experiment. Believe me, it's by far the best way to learn about programming.

Now we were wondering how the micro would react to END instead of end. Right then, just type in:

END [Enter]

(remembering that [Enter] means press the Enter key, don't type in Enter ...). As you'll see, the micro doesn't throw it out.

In fact the Amstrad doesn't mind if you type in your keywords in upper or lower case (unlike some other, more pedantic micros that demand keywords in upper case only.)

Another thing you've probably realized is that the keys have an auto-repeat facility. This means that having pressed a key, if you keep it down the letters repeatedly print themselves out automatically.

Press Enter to get on a new line,

ignoring any System error messages. Using auto-repeat, press several letters on a line but don't press Enter. Right, you should now have a line of characters with the cursor hanging on the end of them.

Press the key marked Del — you'll see that the character on the left of the cursor is "devoured" by it.

This is one way of getting rid of, or deleting, characters (hence the Del). If you keep Del pressed down it will auto-repeat and gobble up the whole line, making a plaintive beeping sound when there's nothing left for it to eat!

You can use Del in this way to correct typing errors — just delete back to the mistake and retype.

Doing this sort of computer correcting is known as editing. There are more sophisticated ways of doing it that we'll be covering, involving the Ctrl, Copy and arrow keys. We'll leave dealing with those for another time.

By now, with all this experimenting, you'll have probably filled up a screenful of text and seen the scrolling action demonstrated. If not, press [Enter] several times in succession.

As you'll soon see, scrolling is when the top of the screen rolls up to allow more typing at the bottom.

If you're wondering why I didn't say just hold Enter down to get the auto-repeat — it's because Enter doesn't auto-repeat.)

If you've had the patience to keep on pressing Enter until everything disappeared off the top of the screen ("scrolled off" is the jargon) you'll have a blank screen with the cursor at the bottom. There are easier ways of clearing the screen, however. Type in

some garbage and then follow up with:

cls [Enter]

The screen should clear, and you'll be left with the ready prompt and the cursor at the top of the screen. If you are not convinced about the CPC464 accepting keywords in both upper and lower case, try:

CLS [Enter]

Both are equally effective.

Well, this has been just a brief examination of the keyboard. There's lots more to cover, including the Esc and Ctrl keys. For the moment though, we'll change tack — after all, it's a computer, so let's get it to compute!

Don't worry, though — this isn't going to turn into a mathematical treatise. After a brief but necessary foray into simple sums this series is thoroughly non-mathematical.

Before we start, let me give you a warning. The computer will do exactly as you tell it, but only what you tell it.

It's a very literal machine and in this respect is like my daughter on a particularly mischievous day. When asked to put on her pyjamas for bed she did exactly as she was told.

Of course, I hadn't asked her to take off her clothes first, had I? You can imagine the results.

Similar things happen with the computer. Say we wanted the micro to calculate $2 + 2$. Not only do we want it to do the sum, we also want it to tell us the answer when it's finished.

We instruct the Amstrad to write things on the screen with the Basic keyword print. This is a relic from the days when the computer's output, as

If you keep Del pressed down it will gobble up the whole line, making a plaintive beeping sound when there's nothing left for it to eat!

It's called, was actually printed out on paper rather than a screen as it is mine.

So, to see the answer to $2 + 2$, type:

print 2+2 [Enter]

Note that you don't need the "=" sign as you do on a calculator. [Enter] takes care of that. Before continuing with this article, why not try a few simple addition sums? Also, try using **PRINT** instead of **print**.

Just as the micro does not allow you to use O or a for 0, so it does not let you use x or X for multiply. You must use the * symbol instead. For example try:

PRINT 4*3 [Enter]

Minus is straightforward. You'll find it sharing a key with =. Divide, however, is not + but an oblique stroke /. For example $12 \div 4$ becomes

PRINT 12/4 [Enter]

Though this may seem a little odd at first, you have met it when dealing with fractions: $3 \div 4$ is equivalent to the fraction $\frac{3}{4}$. Try:

PRINT 3/4 [Enter]

From now on I am going to assume you accept that before the CPC464 can act on your instructions they must be sent to it by [Enter]. I will, therefore, omit [Enter] from my examples. Make sure you don't!

Before experimenting with further some of your own devising, I'd like you to try the following sequence:

**PRINT 3+8-3
PRINT 4*2
PRINT 4*8-2
PRINT 4/8*2**

If you think carefully about the results you'll see that the computer interprets sequences of sums in the order you learned at school. You do whatever is inside brackets first, multiplication and division next, then finally addition and subtraction.

Now try:

**PRINT 3/3
PRINT 1000*1000*1000
PRINT 1/100/100/100/100/100**

If you've managed this correctly, you should get:



The point to note here is that the micro works to a limit of accuracy. For example, $2/3$ is not exactly 0.66666667. The error is well under a millionth, though. Still, it must be borne in mind.

Similarly with especially large or small numbers. The computer saves space by storing them using a scientific notation called Exponent format. Here, for example, instead of printing out the answer to

1000*1000*1000

as you might expect:

1000000000

it prints out the result as 1E9.

For E, which stands for Exponent, you should read "multiplied by 10 to the power of". For example, 1E9 means "1 multiplied by 10 to the power of 9" which, if your maths is up to it, gives you the correct answer.

Similarly, the answer for

1/100/100/100/100/100

was returned as 1E-10 which reads as "1 multiplied by 10 to the power of -10" which is 0.0000000001, the correct answer.

If you don't follow all this, don't worry. I've only covered it in this article to warn you about odd looking results to your sums which might pop up and confuse you.

Now let's try to get the computer to print out some words. Let's get it to print out Hello.

If you cast your mind back to your schooldays (and for some of us that's an awful long throw), you'll remember that when someone says something you surround what that person says with quotation marks for quotes for short, such as, He said, "Hello".

In Basic, of course, we don't say words, we **PRINT** them, but we still

surround them by quotes. We omit, however, the comma and full stop. Try:

PRINT "Hello"

The micro should print out:

Hello

Notice that the quotes are not printed. So, to get the CPC464 to print out a message on its screen, we just use **PRINT** or **print** followed by the message surrounded by quotes.

The message in quotes is called a string, or a string literal. "String" is because the micro considers them just to be a string of letters, one after the other. "Literal" is because the micro prints out literally, or exactly, what is between the quotes. So:

**PRINT "Hello"
PRINT " "Hello"
PRINT " "Hello"**

give different outputs since in each different number of spaces precede the Hello.

Actually strings do not have to be words. They can be any combination of symbols, including numbers. Just keep them in quotes. Try the following:

**PRINT "4*3"
PRINT 4*3**

This should convince you that the computer does print out strings - that is, what's between the quotes - literally.

When the calculation is in quotes the computer simply echoes the sum on the screen. When the calculation is not in quotes, it prints out the answer.

Experiment with printing out various messages on the screen. How long can you make them?

At the moment, the micro is responding to our commands as soon as we send them by pressing [Enter], but in a calculation or task requiring several steps this can be rather tedious.

It would be more satisfactory to give the computer a whole series of instructions that it could get on with, rather than spoon-feed it step by step.

Such a sequence of instructions is called a program, and we shall begin writing programs in next month's installment.

AMSTRAD AT LION HOUSE



BRITAIN'S LARGEST AMSTRAD CENTRE



Main stockists of all Software and
Peripherals for the Amstrad CPC464.

Business, Educational and Games
Software, Joysticks, Power Supplies,
Printers and Books.



Lion House,
227 Tottenham Court Road,
London W1P 0PS.

Telephone 01-580 7285.

ER*BERT'S 'ERE For the AMSTRAD CPC 464

IT'S ER*BERT'S CUBIC DOMAIN
FAST - FUNNY - ADDICTIVE!



- Avoid his unwelcome guests.
- Grab the bananas - double your score! - but watch out for Boris he will soon want it back! Drop it and run, unless you are very brave!

- Avoid cascading balls and the rising hole - don't let Gulp the antacid give you a bubble you'll never forget!

- Escape when it gets really tough by transporter drop or roll that - but only if you've earned one.

- Multiple screens - additional cube colour changing Mobs.

- It's fun at Level One - but watch out at Level Ten!! Packed with fun and excitement.

ER*BERT Machine code game £5.95

includes vial and strategy

000000 HI:003100
HATS G



MICROBYTE SOFTWARE (UK)
18 KINGSTON ROAD, READING, OXFORDSHIRE, RG1 1NE.

**MICROBYTE
SOFTWARE**

Available NOW at
some retailers - or by
fast mail order direct
from Microbyte
Software.

Order your
copy today for
£5.95 only.

ORDER ONLY WE DON'T
DELIVER



COMING SOON...
3D SPACE RANGER

The Space game to
challenge your skill!

AVAILABLE FOR AMSTRAD
JANUARY 1985





Grab your paper pen and ink...

... you're going to explore the colourful world of Amstrad graphics with the help of **MICHAEL NOELS**

WELCOME to the colourful world of the Amstrad CPC484, and congratulations on having such a superb machine for graphics programming.

If you've run any commercial arcade games, or typed in the programs from this issue, you've probably already seen the amazing graphics effects the CPC484 is capable of.

However because of the Amstrad's wide range of graphics and colour commands, incorporating them into your own programs can be a little difficult at first — and the User's Instructions aren't too helpful.

So here's a gentle-paced, no-nonsense introduction to the ins and outs of graphics and colour programming that you won't need a PhD in computer science to understand.

I have assumed that you know a little BASIC, but don't worry if you don't — you can pick it up as you go along. And if you don't happen to have a colour monitor you can still

take advantage of the techniques we'll be using.

So switch on your Amstrad, or reset it if it's already on. To reset it press the **Esc** key while holding **Ctrl** and **Shift** down. You should see a blue screen with the familiar message appearing in yellow writing.

Let's see how many characters you can fit onto one line of the screen. Type:

0123456789

repeatedly, and you'll find that the screen has a width of exactly 40 characters.

You can actually have three basic types of screen — these screen types are known as modes — all with different screen widths. Enter:

mode 0

and see how many characters wide the screen is now. Incidentally, if you've got a syntax error here, you've probably missed the space *not* between *mode* and *0*!

As you'll have discovered, *mode 0* has 20 characters — rather far from 40 that. Now try:

mode 1

Again the screen is cleared, but this time we get a "bikini" shape. If you're up to all the typing, you'll find that you can fit 90 characters across the screen.

When you first switch on or reset you are in *Mode 1*. Prove it now by entering:

mode 1

As you can see, we're back to normal size, and can fit 40 characters across the screen.

So we've got three modes — 0, 1 and 2. Notice it's not 1, 2 and 3. Remember, computers always start

their counting at zero, not one.

The number of characters across the screen is not the only difference between modes – they also differ in the number of colours they allow on the screen at once.

Mode 0 allows 16 colours, Mode 1 four and Mode 2 permits two colours. Notice that the more colours you have the less characters you get across the screen, and vice versa.

When you think about it, it makes sense. You've only got a fixed amount of memory reserved for the screen, so if you're keeping track of a lot of colours you've not got much spare for remembering a lot of characters.

On the other hand, if you decrease the number of colours you've got to remember, there's more memory space available to keep tabs on a larger number of characters. Table 1 summarises it.

Mode	No. of characters	No. of colours
0	256	16
1	40	4
2	80	2

Table 1: Mode characteristics

If you've been following so far, you should be in Mode 1. If not reset and we'll return to writing in yellow on a blue background, with a screen width of 40 characters.

Looking at it logically the smallest number of colours you can have in a mode is two. If you're going to see anything on the screen at all you'll need a foreground colour and a background colour.

For instance, in order for you to see the writing on this page, we've

chosen black to be the foreground colour (that is, the colour that prints a period in) and white for the background (the colour of the paper).

Of course our printers could swap this round – and sometimes do – so that the letters appear in white on a black background, giving a sort of negative.

If we really wanted to go berserk we could print it in a white foreground on a white background, only you wouldn't be able to see it because of the lack of contrast. Oddly enough, this sometimes comes in handy on the Amstrad!

At the moment as far as the CPC464 is concerned I want you to imagine that we're writing on blue paper with a pen filled with yellow coloured ink.

All right, it's come clean – the reason for the tedious last sentence was that pen, ink and paper are special words as far as the Amstrad is concerned. Enter:

pen 1

and you'll see the Ready prompt as usual after a direct command, but it's changed colour. Now it appears in cyan.

If you try typing in a few characters – it doesn't matter which – they should all appear in cyan, though still on a blue background.

Next, try:

pen 2

and the writing should now appear in red. Then enter:

pen 3

and our characters will appear in yellow again.

Great! We've got three pens to write with, have we? No. We've actually got four – each filled with a different coloured ink – and, as per usual computer practice, we number them from 0 to 3.

If we want to change pens, we simply type pen followed by a space and then the number of the pen we want. So:

pen 0

puts out writing in bright red. Type:

pen 0

and try some writing. You won't be able to see a thing because – if you haven't guessed yet – pen 0 is blue so

you're writing in blue ink on blue paper.

So how do we get out of it? Well, press Enter to get you on a new line, then carefully type:

pen 1

and press Enter again. You should get back to yellow writing.

If you can't manage this – and it can be really awkward typing when you can't see what you're doing – reset the machine. Now try:

pen 4

Sorry about that! You're writing in blue on blue again. Oh well, at least you know how to get out of it this time – and you know that pen 4 is the same as pen 0. Reset your machine and try:

pen 2

The writing's still in yellow. So pen 2 is the same as pen 1. So what about pen 3? Try it:

pen 4

We're in cyan, the same as pen 2. No prizes for guessing that:

pen 7

gives you red. Let's explain: When you switch on or reset, you are in Mode 1. Now Mode 1 only allows four colours or inks on the screen at once. So when you've gone from pen 0 to pen 2 the CPC464 starts again by making pen 4 equivalent to pen 0 and so on. Similarly pen 6 is equivalent to pen 0, and so on.

So in Mode 1, given a pen number, it's equivalent to the remainder left when that pen number is divided by 4 (since it's a four colour mode). Hence pen 13 is equivalent to pen 1. I always imagine that the pen numbers are "wrapping round" to the start again.

If the above maths has you fazed don't worry. If you don't try anything fancy, and stick to the numbers 0 to 3 for your Mode 1 pens, you'll be all right. Table 2 summarises the colours

Pen number	Colour
0	Bright blue
1	Bright yellow
2	Bright cyan
3	Bright red

Table 2: Default colours in Mode 1



associated with the pen numbers.

Now run Program 1, which illustrates the different colours available. You could add the following lines to illustrate pen 0:

```
10 pen 0
20 PRINT "This is in pen 0"
```

but, of course, you wouldn't see it.

```
10 RUN PROGRAM 1
20 MODE 1
30 PEN 1
40 PRINT "This is in pen 1"
50 PEN 2
60 PRINT "This is in pen 2"
70 PEN 3
80 PRINT "This is in pen 3"
```

What would happen if we ran it in Mode 2? Change line 20 to:

```
20 mode 2
```

and see.

What happens to pen 2, and why's pen 3 yellow? It used to be red! Well Mode 2 is a two colour mode. Red gives us bright blue, pen 2 gives us bright yellow – then we're run out of available colours, so we start again as we did when we ran out of colours in Mode 1 (only then there were four available).

So pen 2 wraps round to blue and disappears against the blue background, while pen 3 becomes yellow, and so on. Table III shows the colours associated with the pen numbers in Mode 2.

If we change line 20 to:

```
20 mode 0
```

there seems to be little difference from when you ran it in Mode 1, save for the latter characters. Don't forget, though, this is a 16 colour mode – our pens should go from 0 to 15.

Program 1 illustrates the idea – showing all 16 colours – including the rather natty flashing colours of pens 14 and 15. Table IV shows the

```
10 RUN PROGRAM 1
20 MODE 0
30 PEN colour = 0 TO 15
40 PEN colour
50 PRINT "This is colour 'colour'"
60 NEXT colour
```

Pen number	Colour
0	Bright blue
1	Bright yellow

Table III: Default pen colours in Mode 2

colours associated with the pen numbers in Mode 0.

Try changing line 20 to give Modes 1 and 2 and you'll see how in modes with less colours the pen numbers wrap around. If you saw error pen 16:

Isoper argument

will be fixed back at 0. The CPC464 knows that the biggest pen number it can possibly have is 15, so it throws pen 16 out. In Modes 1 and 2, as we've seen, it wraps the pen numbers round, but it still rejects numbers over 15.

So far all our work has been done on a nice blue background, but we aren't restricted to this. Let's investigate.

Reset your micro so you are back in Mode 1. Now so far we've been writing with pens filled with different coloured inks – on blue paper. Enter this:

```
paper 1
```

All of a sudden Reedy appears on red paper. That is, the letters still appear in yellow, but on a red background. You see:

```
paper 2
```

Pen number	Colour
0	Bright blue
1	Bright yellow
2	Bright cyan
3	Bright red
4	Bright white
5	Black
6	Bright blue
7	Bright magenta
8	Cyan
9	Yellow
10	Pastel blue
11	Pink
12	Bright green
13	Pastel green
14	Flashing blue
	Flashing yellow
15	Flashing pink/sky blue

Table IV: Default pen colours in Mode 0



means "write on paper that's the same colour as the ink in pen 3".

Now, from Table I, pen 3 is bright red, so paper 3 sets the background to bright red. Type in some characters of your own if you don't believe me. Next try:

```
paper 2
```

The ink in pen 2 is bright cyan, so our writing now appears on cyan paper. I find this terrible difficult to read, so let's make it clearer by

changing our foreground colour to red. Remember how? It's

pen 3

Paper colour is really quite easy to use – it works just as pen does, and follows the same restrictions as to mode. Just bear in mind, paper colour means the background colour is that of the ink in pen 0.

Notice that so far only the background of the characters you've typed has been in the new paper – the rest of the line stays in the old paper. When you've reached the bottom line, however, and the new line scrolls up, the whole of that line will be in the new paper.

After all, it's got to be in something, and as it's brand spanking new we may as well have it in the new paper.

There is a quicker way to get the screen in the new background colour. Enter:

paper 1 : 010

and the screen will clear to a yellow background (paper 1) with writing in the old background colour (we've still in pen 0).

You'll also notice something else if you haven't already – our yellow paper is surrounded by a blue border. You haven't noticed it before because our background's always been blue, matching the border.

We'll see later how you can change the border's colour. In fact there's not much else you can do with it – we can't actually write anything there...

Before we continue, have a look at Program 18, which illustrates how

the various pen and paper combinations work.

So far we've only seen 16 colours. However, when we bought your Amstrad you were promised 27. What's happened to them?

Program 19 shows where they've been hiding. It successively steps the border through all 27 colours of ink, as they are known.

```
10 REM PROGRAM 19
20 MODE 1
30 FOR colour = 0 TO 26
40 BORDER colour
50 LOCATE 14,13
60 PRINT "Border '1' colour"
70 FOR delay = 0 TO 999 :NEXT delay
80 NEXT colour
```

As you'll have guessed border is the command that changes the colour of the border – you simply follow it with a space and the number of the colour you want the surround to be.

But beware, these numbers won't appear to have anything to do with the numbers you've been using for pen and paper. For example:

border 0

The border turns black – not blue as you would expect from pen 0 and paper 0.

This is a very important point – the numbers used with pen don't label colours – they label pens, which just happen to be filled with "coloured ink".

Just because pen 3 has so far always given us red in Mode 1, it doesn't have to. It's just that, at the moment, pen 3 happens to be filled with red ink.

Later on I'll show you how to fill a pen with, say, blue ink – in fact any coloured ink from our "palette" of 27 colours.

So the 3 in pen 3 labels the pen, not the colour of the ink it is filled with. As in Mode 1 we've allowed four pens, and hence four corresponding papers. We can fill these pens with any four of the 27 – a sort of "pens and 4 from 27". In fact you can fill all four pens with the same colour if you want.

Much the same holds for the other modes, with their different number of pens.

Now the main needs some way of referring to each of the 27 available colours. It could, of course, do it in



words – orange, bright red and so on.

Being a computer, it prefers to give the various coloured ink reference numbers, as shown in Table V.

As you can see, ink 0 is black and as the border command uses the ink number NOT the pen number:

border 0

turns the border black – and leaves the screen entirely alone. You must use pen or paper to affect the screen.

Next month I'll show you how to fill the pens with any ink you care to choose. For now, though, it's probably best if you just use the ink that the pens are "supplied" with when you switch on or reset – the default ink as they are known. Tables 11, 12 and 19 show them for each mode.

That should keep you busy enough until our next issue!

Ink number	Colour
0	Black
1	Blue
2	Bright blue
3	Red
4	Magenta
5	Mauve
6	Bright red
7	Purple
8	Bright magenta
9	Green
10	Cyan
11	Sky blue
12	Yellow
13	White
14	Pastel blue
15	Orange
16	Pink
17	Pastel magenta
18	Bright green
19	Sea green
20	Bright cyan
21	Light green
22	Pastel green
23	Pastel cyan
24	Bright yellow
25	Pastel yellow
26	Bright white

Table V: Ink colours

```
10 REM PROGRAM 11
20 MODE 0
30 FOR background = 0 TO 13
40 PAPER background
50 CL0
60 PRINT "This is paper '1' background"
70 PRINT
80 FOR colour = 0 TO 13
90 FOR pen colour
100 PRINT "This is colour 'colour'"
110 NEXT colour
120 PRINT "Press any key"
130 delay = 10000 : 30 delay** 10
140 NEXT 130
150 NEXT background
160 PAPER 0 : PEN 1
```

You're never too young to play a Magical Adventure on the Amstrad CPC464...

Based on the style of the classic computer adventures – but written so that even small children can learn to find their way around, encouraged by colourful graphics and exciting sound effects.

The pack contains a 48-page full colour storybook

PLUS

a full length multi-location adventure on cassette for only

£8.95! post free

**Read the book
– then play
the game!**



**Ideal
Christmas
present**

Please send me the complete Magic Sword pack for the Amstrad CPC464 (containing storybook and cassette to)

Name

Address

- ☐ I enclose my cheque for £8.95 payable to Database Publications
☐ Or debit my Access/Visa card:

No.

Signed

SEND TO: Adventure offer, Europa House, 58 Chester Road, Hazel Grove, Stockport SK7 5NY

TAKE A TOUR ROUND YOUR MICRO'S SOUND

First in an
informative series
by NIGEL PETERS

MOST micro newbies have the ability to make sounds. The simplest way to get your CPC464 to make a noise is to enter:

PRINT CORNED

and press Enter. The trouble is that this solitary beep isn't all that exciting.

To enable you to create rather more interesting sounds the Amstrad has three very sophisticated commands - **SOUND**, **ENT** and **ENV**. However, the more sophisticated the command the more difficult it is to use, at least at first.

So over the next few months I will be covering the commands in detail, building up a step-by-step guide to making noises on the Amstrad CPC464. It will be a practical course with lots of examples, so make sure that your micro is switched on and ready for use while you read the articles.

And don't take everything I say on trust - try it out. The **SOUND** command is a big subject and only by trying things out for yourself will you grasp it in its entirety.

Having said all that, let's make a sound. Enter:

SOUND 1,200,100,7

and feel yourself thrill to the exciting

tones of your Amstrad.

Well, maybe I am overdoing it a bit. It's not the most exciting sound in the world. Still, once you understand how that **SOUND** command works you'll have gone a long way to conquering Amstrad sound.

The simplest form of the **SOUND** command is followed by four parameters. Don't be put off by parameters, they are just the numbers that follow the **SOUND** command and affect the way it works.

The parameters in the last sound we made are 1, 200, 100 and 7. The first number (1) selects the channel that the note will be played on, the second (200) tells the micro what pitch it will be. The third parameter (100) determines how long that note will last, while the last figure (7) decides on the volume it will be played at.

If you enter:

SOUND 1,200,100,7

we produce a note on channel 1 which will last for one second, be played at full volume and be fairly high pitched.

Don't worry if you don't fully understand all these terms, we'll be dealing with them later in this article.

The basic structure of the **SOUND**

command is:

SOUND channel,at,for,level,volume

that is, the Basic keyword **SOUND** followed by four parameters, which specify the particular sound wanted.

Have a go at a few more sounds by entering:

SOUND 2,100,20,4

or:

SOUND 4,50,50,1

or be adventurous and make up your own.

As you'll see (and hear from the above), the values following the **SOUND** command can vary and as they vary so do the notes produced. However you can't just pick any old parameter, you can only choose between certain limits.

Take the channel parameter, the first number that comes after the **SOUND** command. This is used to decide which of the CPC464's sound channels is to be used. The Amstrad has three such channels, and each can only play one note at a time.

However by having all three channels working at the same time you can have the micro producing three different notes simultaneously. The channels are known as channels A, B and C and the parameters that

select them are 1, 2 and 4 respectively.

Program I shows these parameters in action. First of all it plays a note on channel A (parameter 1), then waits for you to press a key. The next note is played on channel B as the parameter following the SOUND command is the figure 2.

```
10 REM PROGRAM I
20 REM Channel A
30 SOUND 1,470,100,7
40 WHILE INKEY=""GOTO 10
50 REM Channel B
60 SOUND 2,370,100,7
70 WHILE INKEY=""GOTO 50
80 REM Channel C
90 SOUND 4,300,100,7
```

Program I

When this note is over and you press another key, the last note, with channel parameter 4, plays on channel C.

Of course these notes were played one after another, I hope the more sceptical of you will be thinking: "How do I know there are three channels? There could be just one channel with the notes playing on it, one after another".

If you do have such doubts, have a go at Program II which plays the same notes at the same time, on different channels. Not exactly Mozart is it? Still, it should have convinced you about the three channels.

Now let's move on from the channel parameter - we'll be back to A in a later article - and look at the pitch parameter that follows it. As you might guess, this controls the pitch of the note produced by the SOUND command.

The pitch of a note is how high or low it is. The higher the note - like a soprano or a mellowed cat - the higher the pitch is. The lower the note

- like a bass or a loghorn - the lower the pitch.

Run Program III and you'll hear a note that changes pitch, going lower and lower. Press ESC a couple of times when you've had enough.

What's happened is that each time round the FOR ... NEXT loop the SOUND command of line 30 makes a note. However each time round the loop the value of the pitch parameter, pitch, varies. At first it's 33, then it's 32, then 34 and so on. As the pitch parameter varies so does the note produced.

```
10 REM PROGRAM III
20 FOR pitch=32 TO 40
30 SOUND 1,pitch,100,7
40 NEXT pitch
```

Program II

You might have noticed that as the pitch parameter gets larger in value, the note gets lower in pitch (the sound gets deeper). This means that a pitch parameter of 100 produces a much lower note than a pitch parameter of 30.

Incidentally if you did use ESC to get out of the previous program you might have left a few notes still in the notes. These may cause the following programs to sound odd for, rather, odder so get rid of them by entering:

```
SOUND 127,100,0,0
```

I know that the channel parameter looks wrong, but all will be explained in a later article. Just use it for garbage collection for the time being.

To return to the pitch parameter, Program IV uses a step of -4 to decrease the pitch parameter each time round the FOR ... NEXT loop. The result is a note that increases in pitch.

The pitch parameter can take values that range from 0 to 4095 though it does get a bit ragged at the extremes of its range. The numbers have to be integers (whole numbers). If you use a number with a decimal part for the pitch parameter the CPC just ignores the decimal.

Don't just believe me, try it for yourself, is there a difference between:

```
SOUND 1,100.5,100,7
```

and

```
SOUND 1,100,100,7
```

that you can detect?

For those with musical pretensions

```
10 REM PROGRAM IV
20 FOR pitch=470 TO 250 STEP -4
30 SOUND 1,pitch,100,7
40 NEXT pitch
```

Program IV

Appendix VII of the User Instructions gives the full range of pitch parameters with their corresponding musical notes. They tell the pitch parameter the time period, but they're the same thing.

Table 1 shows 24 of these notes and their pitch parameter values.

Program V plays a selection of these notes in order, reading the values of pitch from the DATA statements of line 50.

```
10 REM PROGRAM V
20 FOR notes=10 TO 13
30 REM pitch
40 SOUND 1,pitch,100,7
50 NEXT notes
60 DATA 119,113,106,100,95,89
70 REM 84,80,75,71,67,63
```

Program V

You might notice that this series of notes has a more "complete" feel about it than the previous ones we've had. We'll come back to this in a later article.

For the moment let's press on to the next parameter which determines

Pitch		Pitch	
NOTE	parameter	NOTE	parameter
C	129	F#	64
C#	125	G	60
D	113	G#	55
D#	106	A	51
E	100	A#	47
F	95	B	43
F#	89	C	40
G	84	C#	36
G#	80	D	33
A	75	D#	29
A#	71	E	27
B	67	F	25
C	63	F#	23
C#	60	G	20
D	56	G#	18
D#	53	A	16
E	51	A#	14
F	47	B	12

Table 1

TAKE A TOUR ROUND YOUR MICRO'S SOUND



how long the note is going to last. This third parameter, controlling duration, can have values from -32768 to 32767 but for the present we'll just use the values 0 to 32767. This number represents the number of 1/100ths of a second the note is to last.

So if the duration parameter is 100 the note should last for one second. If the duration parameter is 1000 it should last for 10 seconds. Enter:

```
SOUND 1,200,300,7
```

and you'll find the note lasts for three seconds.

Program VI, a variant of the previous one, uses the familiar FOR ... NEXT loop to vary the duration parameter via the variable *duration*.

```
10 REM PROGRAM VI
20 FOR duration=10 TO 100 STEP 10
30 READ pitch
40 SOUND 1,pitch,duration,7
50 NEXT duration
60 DATA 118,115,106,100,95,89
70 DATA 84,80,75,71,67,63
```

Program VI

As the notes change in pitch each time round the loop they also increase in length from 1/10th of a second to 1.2 seconds. This gives a sort of slowing down effect.

Program VII decreases the dur-

ation parameter each time round the loop, so giving the opposite effect.

The final parameter we're going to deal with is the volume parameter. This, as is obvious from the name, decides how loud the note is going to be and can have whole number

```
10 REM PROGRAM VII
20 FOR duration=100 TO 10 STEP -10
30 READ pitch
40 SOUND 1,pitch,duration,7
50 NEXT duration
60 DATA 118,115,106,100,95,89
70 DATA 84,80,75,71,67,63
```

Program VII

values from 0 to 7. If the parameter is 0, the note is at its quietest. The parameter 7 is a little louder and so on to maximum volume when the volume parameter is equal to 7. Program VIII shows them in action.

```
10 REM PROGRAM VIII
20 FOR volume=1 TO 7
30 SOUND 1,100,100,volume
40 SOUND 1,100,100,0
50 NEXT volume
60 DATA 118,115,106,100,95,89
70 DATA 84,80,75,71,67,63
```

Program VIII

The main part of the program is easy enough to follow. As the FOR ...

NEXT loop cycles the volume parameter increases and the note gets louder. But what of line 40 which has a SOUND command with a volume parameter of 0? What's that doing there?

The answer is that it's there to provide an interval of no sound between the other notes. When the volume parameter has a value of 0 it means that the sound has no loudness at all. You can't hear it.

However, the note still plays (albeit silently) for the full length of the duration parameter. This means that line 40 produces one second silences between the notes.

Although using a 0 volume parameter to produce a note that you can't hear seems a little strange at first sight, it is useful. You can use it to produce the rests that are often found in tunes. Also it makes things a lot clearer at times. Leave out line 40 in the last program and you'll see what I mean.

And that's the end of our tour of the four SOUND parameters, except for one thing. Can you explain why the notes produced by:

```
SOUND 1,100,20,4
```

and

```
SOUND 1,200
```

sound exactly the same?

The reason is that if we don't supply the SOUND command with a duration or volume parameter it provides them for itself. The default duration parameter is 20 and the default volume is 4, hence the two SOUND commands above produce the same note.

And that's it for this month. Table II sums up what we've covered so far, while Program IX makes use of it to produce a familiar tune.

Why not write a few simple tunes yourself?

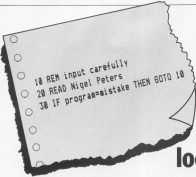
```
10 REM PROGRAM IX
20 SOUND 1,220,100,7
30 SOUND 1,215,100,7
40 SOUND 1,200,100,7
50 SOUND 1,250,100,7
60 SOUND 1,230,100,7
```

Program IX

■ **Next month:** More exploration of the sound capabilities of the Amstrad CPC464.

	Channel	Pitch	Duration	Volume
range	1 = A, 2 = B 4 = C	0 to 4095	1 to 32767	0 to 7
default	none	none	20	4

Table II: Parameter ranges for SOUND commands



```
10 REM input carefully
20 READ Nigel Peters
30 IF program=mistake THEN GOTO 10
```

How to avoid those listings loopholes

ONE of the good things about this magazine is that it's going to be full of listings. Long ones, short ones, simple ones and hard ones — they'll all be there, waiting for you to type them into your CPC464.

This, however, is sometimes easier said than done. If you're not careful or are just plain unlucky, typing in listings can be extremely frustrating.

I'm sure we've all had the experience at one time or another of typing something in and finding that it doesn't work. And no matter how hard we try, the fault is untraceable. If you haven't had this happen to you yet, touch wood, quickly.

Eventually we give up in disgust and say that the listing is wrong. Usually in fairly strong language.

To be fair, this can be the case, though it very rarely is. Occasionally a part of a listing won't print out properly or a space will appear where there wasn't one in the first place. It's even been known for part of a listing to "fall off" a page.

However this doesn't happen all that often (though once is one time too many). The sad fact is that if the program won't work you've almost certainly made an error typing the

listing into your mine. There's no one else to blame.

I speak from experience (because at one time or another I've made every error possible (and some that I'm sure aren't). Even now, after hundreds of hours at a keyboard, when I should know better I still end up looking at a listing convinced that I've not made a mistake. But, in my heart of hearts, I know that the odds are that I have.

So from my vast experience of typing in listings wrongly I'll give you a few hints on how to stop these errors creeping in.

The first thing is don't get carried away. It's all too easy to become obsessed with a listing that's gone wrong, struggling with it for hours after hours, eventually losing your temper.

If you don't spot the mistake in the first quarter of an hour then give up for a little while. Go and have a cup of coffee or something. See if the family still recognises you. It's surprising how often you see the mistake as soon as you try again after a rest.

So don't get too involved. It doesn't help. And remember, we do it for fun, our lives don't depend on it.

One of the daffiest mistakes you can make is to leave a complete line out. Of course the program won't

work properly if it isn't all there, but lots of times we expect it to.

Often we just miss the line, skipping it by accident as we read the listing. Alternatively, instead of typing in two lines, 40 and 50, we type in the first line as line 40 and then number the next one as 40. The second line overwrites the first and so we have a missing line. To make it worse, the line numbered 40 is actually the one that should be numbered 50. It's amazing how difficult this can be to spot.

And the trouble is that the Amstrad can't tell you that "line 50 is missing" because (unless it's used in a GOTO or GOSUB) it doesn't know that it should be there. So the error message that it comes up with isn't all that helpful, often pointing to some other part of the program that depended on the missing line.

Suppose line 50 was something like:

```
cowboys=8
```

that is, it gives the variable *cowboys* a value of eight. Now if you leave this out the micro will either use an earlier value of *cowboys*, which might be completely wrong, or assume, since it

hasn't come across it before, that it's equal to zero.

Either way it won't tell you that there's a line missing. You have to hunt through and find it for yourself.

The moral is that when you're typing in a listing, type in all of it. And type it in exactly. There's no point in changing the programmer's code and then feeling aggrieved when it doesn't work.

Yet people do this, correcting "mistakes" that they find in listings and then wondering why the program prints to a halt 20 lines later.

So type in the listings exactly – and watch your spelling, because the Amstrad is very sensitive about such things. It can only understand a few words, the Basic keywords such as PRINT, LIST and LET.

Misspell these and you're in trouble, as Program I shows. Happily, you are told if you've done this, so it's not too hard to look down.

```
10 REM Program I
20 LET indiana=0
30 PRINT "There are 'indians'
  indians"
```

Of course, you wouldn't do anything so daft, would you?

And it's not just keywords where spelling is important. Mispell a variable and the micro can easily get confused. Look at Program II.

```
10 REM Program II
20 LET indiana=0
30 PRINT "There are 'indians'
  indians"
```

The micro comes across *indians*, can't find any reference to it and gives it a value of zero. In this case it's immediately apparent that there's been an error, but in long listings these "ghost" variables can be very difficult to spot.

So if you want a listing to work, you have to type it in correctly. This is easier said than done, as it's amazing how your mind can wander as you're doing it.

I've found that it makes life a lot easier if I enter a long listing in three or four sessions rather than one,

saving it to tape between times.

Incidentally, I recommend that you **SAVE** your program every 20 or 30 lines. This is in case you lose your listings by an accident such as a power cut or a middle-aged younger brother. You might lose all your work from the Amstrad's memory but you will have the best part of it on tape.

So far we've looked at what can go wrong if we mispell Basic keywords and variable names. Now let's look at a classic error that I'm sure everyone has made at one time or another. This is caused by mistaking a number for a letter and vice versa.

Probably the most common, and the hardest to detect, is confusion between the numeral "0" and the lower case letter "o".

On poor quality listings these look very much the same, and even with clear listings it's all too easy to press one key in mistake for the other. Even with a slash across the 0 this can happen.

The Amstrad won't like it and the program will almost certainly grind to a halt. Try out Program III and see what happens.

```
10 REM PROGRAM III
20 FOR loop=0 TO 10
30 PRINT "0 and o are mixed up"
40 NEXT loop
```

The same kind of confusion can arise between the lower case letter "l" and the number 1. They look fairly similar and as typewriters they are often the same key – but not on the Amstrad. Confuse the two and you're asking for trouble – trouble that's very difficult to spot and sort out.

Another pair of localities to be wary of are the minus sign "-" and the underline "_", which are often confused. You can't do a subtraction with the underline sign, though it's amusing how many times you try!

Now let's move on from the problems caused by similarities between letters and numbers to problems caused by getting the punctuation wrong.

In particular it's all too easy to confuse the fullstop ".", the semicolon ";", the colon ":", and the

comma ",". The trouble is that these can look very similar on listings and mistakes are easily made.

This can cause all sorts of problems from having displays in the wrong place or in the wrong colour to having the program crash without an error message.

Also beware of putting in commas in numbers. You may write 20,000 but the CPC won't like it if you type it in. It prefers the number in the form 20000 – without the comma.

Another problem caused by mixed up punctuation marks comes in data lists, where the items are separated by commas. For example, what should be:

DATA 1,2,3,4,5

might be typed in as:

DATA 1.2.3.4.5

The decimal point in 3.5 has become a comma. This will result in the wrong data being read and the program will run incorrectly, if at all.

What's particularly annoying is that the CPC won't tell you that there is now one item too many in the list. It only displays an error message when it tries to READ from a list with too few items.

This can be very tricky to sort out and if you get some weird happenings in a program it's always a good idea to check that the DATA lines are spot on.

The moral is to be very careful with punctuation marks in listings. They may not mean a lot to you, but they do to the micro. Get one wrong and it can be the devil of a job to find it and remedy it.

From punctuation marks let's turn our attention to spaces and the problems caused by a lack of them.

As you know, Basic uses a special set of words, the keywords, for special purposes. The Amstrad allows us to enter them in lower or uppercase letters. However when we list them they appear as capitals.

These keywords must have a space on either side of them or else the CPC thinks that it is just part of a variable name.

Program IV shows what happens if

the space after the keyword PRINT is missed out.

```
10 RUN PROGRAM 10
20 x=5
30 PRINT
```

Alternatively, you can leave out the space before the keyword as in Program V, so the lesson is always to leave spaces before and after basic keywords.

```
10 RUN PROGRAM V
20 x=5
30 IF x=5 THENPRINT :
```

You may have noticed from the above that the CPC allows you to use keywords in variable names. It's perfectly possible to have a line like

```
10 thenprint=5
```

which gives the variable thenprint the value of 5. My advice is to avoid using variables which contain keywords, it's fraught with difficulties.

With all these possible mistakes it's a wonder that any listings ever get typed in correctly! And on top of all this there are two more simple errors that can also be made.

The first is to use the AUTO command to make the micro print out the line numbers, then forget that you're using it and type in the line number again.

A line number like 20.20 is a dead giveaway that you've made this mistake. I've done it so often now that I never use AUTO. The time I've "saved" not typing in line numbers is used up correcting my mistakes.

Incidentally, you'll find that if you make this mistake 20.20 is treated as line 20. Program VI illustrates this.

```
10 RUN PROGRAM VI
20.20 LET indiana=1
30 PRINT "You've forgot about AUTO."
```

The final error is really stupid but, sadly, all too easy to make. It's done by forgetting that you've already got a program in the Amstrad's memory. You then type in your listing and, if the line numbers of the two programs don't match, they get mixed up.

The last line of Program VII is

1. Copying wrongly
2. Misspelling
3. Confusing similar letters
4. Punctuation errors
5. Not leaving spaces
6. Forgetting AUTO
7. Forgetting old programs

These are the seven deadly sins when it comes to typing in listings

obviously left over from an earlier program and shouldn't be there at all.

```
10 RUN PROGRAM VII
20 LET indiana=1
30 LET coskov=1
30 IF coskov=1 THEN PRINT "Enter at your peril!"
```

And that brings us to the end of our survey of the mistakes that can be made when typing in listings. Don't worry, you won't make all of them in the same program, though I'm sure everyone will make them at one time or another.

Now that you know what they are they should be easier to find and correct. One point to bear in mind is that when you do find a mistake in a line don't think that it's the only one.

If you made one error then your concentration was lacking at that point and there may be another

lurking about on the same line or therabouts. It's amazing how often they come in pairs.

And beware of making another mistake when you "correct" a line. It's very easy to introduce another error as you correct the first one — ask any printer (not ours, of course!)

All in all, it just comes down to looking carefully at what you're typing, both on the page and on the screen. If you spend all your time peering at the keyboard (as most beginner typists do) and don't look at the screen to check your work, mistakes will abound. This seems obvious, yet it's easy to become hypnotised by the keys, typing rapidly but not checking as you go.

So pay attention to what you're doing when you type in listings. A little time spent checking as you actually enter the lines saves a lot of time later.

TRAPPER

TRAPPER is a quick action game in which your aim is to trap the evil maze monster as fast as you can. The only way to do it is to corner him so that he cannot move up, down, left or right.

The maze monster is very intelligent and will escape from the most awkward situations. So be warned. You take the part of a hunter who is controlled by the following keys:

Z - Left
X - Right
/ - Down
] - Up

The monster is afraid of everyone, so colliding with him sends him off in a totally different direction. This can be used to your advantage, as you will see when you play the game.

Happy trapping!



```
10 RANDOMIZE TIME
20 REM Trapper
30 REM By Chris Edwards
40 REM 10
50 MODE 1
60 PAPER WUPR 1
70 LOCATE 12,0
80 PRINT"Trapper!"
90 LOCATE 1,12
100 INPUT"Enter maze type, 1=single 2
    =multiple ",level
110 IF level=1 OR level=2 THEN SOUND
    1,200,50:GOTO 30
120 GOS
130 gos=254:gos=049
140 gos=0:gos=049:gos=140,049:140:1
150 LOCATE 1,CURRENT block
160 LOCATE 1,24:PRINT gos=049
```

```
170 FOR loop=1 TO 33
180 LOCATE 40,loop:PRINT CHR$(141);(2
    84:141)
190 NEXT
200 FOR loop=1 TO level*100
210 LOCATE (INT(loop/10)+30)+2,INT(loop
    /10)+2
220 PRINT CHR$(141)
230 SOUND 1,400-loop,1
240 NEXT
250 gos=10:gos=10
260 gos=25:gos=40
270 gos=049:gos=049 700
280 gos=0:gos=TIME
290 type=INT(loop/10)+1
300 ON type GOTO 210,120,330,340
310 gos=0:gos=049 330
```

```
320 gos=0:gos=049 350
330 gos=0:gos=049 350
340 gos=0:gos=0
350 gos=0:gos=049 350
360 IF gos=0 THEN SOUND 1,200,1:GOTO
    370
370 gos=0:gos=049 350
380 gos=0:gos=049 350
390 gos=0:gos=049 350
400 FOR 2:LOCATE 14,23:PRINT"Time = "
    :INT(loop/100)+1:gos=0
410 gos=0:gos=0
420 IF INT(loop/10)+1 THEN gos=0:gos=0
    4
430 IF INT(loop/10)+1 THEN gos=0:gos=0
    4
440
```

By KEVIN EDWARDS



VARIABLES

block\$	40 block characters used to draw the outside of the maze.
tin	Time taken to trap the monster.
level	Maze level. 1=simple 2=complex.
loop	General purpose loop variable.
me	Hunter's graphic character.
mon	Maze monster's graphic character.
monx	Monster's X coordinate.
mony	Monster's Y coordinate.
mx	Hunter's X coordinate.
my	Hunter's Y coordinate.
px	Pixel colour of screen position last tested.
previous	Time when the current screen started.
rx	Monster's relative change in the X direction.
ry	Monster's relative change in the Y direction.
type	Indicates monster's present direction.

Variables not listed are general purpose ones.

```

440 IF INKEY$="0" THEN my=1:GO TO 44
450 IF INKEY$="0" THEN my=1
460 IF my=0 AND my=0 THEN 300
470 px=127:128:129:130:131:132:133:134:135:136
480 my=1:41
490 IF px=14 THEN 300
500 px=127:128:129:130:131:132:133:134:135:136
510 my=1:41:42:43:44:45:46:47:48:49:50:51:52:53:54:55:56:57:58:59:60:61:62:63:64:65:66:67:68:69:70:71:72:73:74:75:76:77:78:79:80:81:82:83:84:85:86:87:88:89:90:91:92:93:94:95:96:97:98:99:100:101:102:103:104:105:106:107:108:109:110:111:112:113:114:115:116:117:118:119:120:121:122:123:124:125:126:127:128:129:130:131:132:133:134:135:136:137:138:139:140:141:142:143:144:145:146:147:148:149:150:151:152:153:154:155:156:157:158:159:160:161:162:163:164:165:166:167:168:169:170:171:172:173:174:175:176:177:178:179:180:181:182:183:184:185:186:187:188:189:190:191:192:193:194:195:196:197:198:199:200:201:202:203:204:205:206:207:208:209:210:211:212:213:214:215:216:217:218:219:220:221:222:223:224:225:226:227:228:229:230:231:232:233:234:235:236:237:238:239:240:241:242:243:244:245:246:247:248:249:250:251:252:253:254:255:256:257:258:259:260:261:262:263:264:265:266:267:268:269:270:271:272:273:274:275:276:277:278:279:280:281:282:283:284:285:286:287:288:289:290:291:292:293:294:295:296:297:298:299:300:301:302:303:304:305:306:307:308:309:310:311:312:313:314:315:316:317:318:319:320:321:322:323:324:325:326:327:328:329:330:331:332:333:334:335:336:337:338:339:340:341:342:343:344:345:346:347:348:349:350:351:352:353:354:355:356:357:358:359:360:361:362:363:364:365:366:367:368:369:370:371:372:373:374:375:376:377:378:379:380:381:382:383:384:385:386:387:388:389:390:391:392:393:394:395:396:397:398:399:400:401:402:403:404:405:406:407:408:409:410:411:412:413:414:415:416:417:418:419:420:421:422:423:424:425:426:427:428:429:430:431:432:433:434:435:436:437:438:439:440:441:442:443:444:445:446:447:448:449:450:451:452:453:454:455:456:457:458:459:460:461:462:463:464:465:466:467:468:469:470:471:472:473:474:475:476:477:478:479:480:481:482:483:484:485:486:487:488:489:490:491:492:493:494:495:496:497:498:499:500:501:502:503:504:505:506:507:508:509:510:511:512:513:514:515:516:517:518:519:520:521:522:523:524:525:526:527:528:529:530:531:532:533:534:535:536:537:538:539:540:541:542:543:544:545:546:547:548:549:550:551:552:553:554:555:556:557:558:559:560:561:562:563:564:565:566:567:568:569:570:571:572:573:574:575:576:577:578:579:580:581:582:583:584:585:586:587:588:589:590:591:592:593:594:595:596:597:598:599:600:601:602:603:604:605:606:607:608:609:610:611:612:613:614:615:616:617:618:619:620:621:622:623:624:625:626:627:628:629:630:631:632:633:634:635:636:637:638:639:640:641:642:643:644:645:646:647:648:649:650:651:652:653:654:655:656:657:658:659:660:661:662:663:664:665:666:667:668:669:670:671:672:673:674:675:676:677:678:679:680:681:682:683:684:685:686:687:688:689:690:691:692:693:694:695:696:697:698:699:700:701:702:703:704:705:706:707:708:709:710:711:712:713:714:715:716:717:718:719:720:721:722:723:724:725:726:727:728:729:730:731:732:733:734:735:736:737:738:739:740:741:742:743:744:745:746:747:748:749:750:751:752:753:754:755:756:757:758:759:760:761:762:763:764:765:766:767:768:769:770:771:772:773:774:775:776:777:778:779:780:781:782:783:784:785:786:787:788:789:790:791:792:793:794:795:796:797:798:799:800:801:802:803:804:805:806:807:808:809:810:811:812:813:814:815:816:817:818:819:820:821:822:823:824:825:826:827:828:829:830:831:832:833:834:835:836:837:838:839:840:841:842:843:844:845:846:847:848:849:850:851:852:853:854:855:856:857:858:859:860:861:862:863:864:865:866:867:868:869:870:871:872:873:874:875:876:877:878:879:880:881:882:883:884:885:886:887:888:889:890:891:892:893:894:895:896:897:898:899:900:901:902:903:904:905:906:907:908:909:910:911:912:913:914:915:916:917:918:919:920:921:922:923:924:925:926:927:928:929:930:931:932:933:934:935:936:937:938:939:940:941:942:943:944:945:946:947:948:949:950:951:952:953:954:955:956:957:958:959:960:961:962:963:964:965:966:967:968:969:970:971:972:973:974:975:976:977:978:979:980:981:982:983:984:985:986:987:988:989:990:991:992:993:994:995:996:997:998:999:1000

```

```

490 LOCATE my,my:PRINT CHR$(ast):RET
500
300 IF ast=32 THEN PER 1 ELSE PER 2
PER LOCATE ast,my:PRINT CHR$(ast):
RETURN

```



Give your fingers a rest...

All the listings from this month's issue are available on cassette. See our special offer on Page 55.

Ready Reference: Graphics

Get the facts at your fingertips with the first of our ready reference charts

pen/paper	ink	colour
0	1	Bright Blue
1	24	Bright Yellow
2	26	Bright Cyan
3	4	Bright Red
5	28	Bright White
6	8	Black
7	9	Bright Blue
8	2	Bright Magenta
9	6	Cyan
10	18	Yellow
11	12	Pastel Blue
12	24	Pink
13	26	Bright Green
14	28	Pastel Green
15	22	Fluorescing Blue / Bright Yellow
16	14, 24	Fluorescing Pink / Sky Blue
17	16, 22	

Mode 0 defaults

pen/paper	ink	colour
0	1	Bright Blue
1	24	Bright Yellow
2	26	Bright Cyan
3	4	Bright Red







Mode 1 defaults

pen/paper	ink	colour
0	1	Bright Blue
1	24	Bright Yellow

Mode 2 defaults

Mode	graphics coordinates		pixels	
	x axis	y axis	x axis	y axis
0	640	400	140	200
1	640	400	320	200
2	640	400	640	200

Graphics coordinates and pixels

Mode	Character	Pixel
0	10 	2 
1	10 	2 
2	18 	2 

Characters and pixels measured in screen coordinates

ink	colour	ink	colour
0	Black	24	Pastel Blue
1	Blue	25	Orange
2	Bright Blue	26	Pink
3	Red	27	Pastel Magenta
4	Magenta	28	Bright Green
5	Green	29	Sky Green
6	Bright Red	30	Bright Cyan
7	Purple	31	Light Green
8	Bright Magenta	32	Pastel Green
9	Green	33	Pastel Cyan
10	Cyan	34	Bright Yellow
11	Sky Blue	35	Pastel Yellow
12	Yellow	36	Bright White
13	White		

Pink colours

Mode	Number of characters	Number of colours
0	20	16
1	40	4
2	80	2

Mode characteristics



Graphics screen



Test screen Mode 0



Test screen Mode 1



Test screen Mode 2

Super graphics where they do the most good

RIGHT from the start I was impressed. As soon as the initial header of *Galaxia* came up I knew I was in for good graphics—and I wasn't disappointed.

The object of the game, one of a set of high quality releases from Karna, is to get past the alien fleet and dock with your starship.

"Ah", you're probably thinking, "Just another sideways scrolling screen with lots of obstacles you must dodge". And you'd be right—but you'll be missing the point by about as much as you would if you described the Mona Lisa as a snapshot of a woman.

As you hurdle down the screen you encounter 18 types of alien, each behaving noticeably more nastily.

If you're the non-violent kind you can move your ship up and down to avoid them. If you're like me you'll blast them by obtaining with your fire button (my psychiatrist says it's good for me—mind you, he's got a *Spectre*).

There are Hammer, Machine, Spider, Ganger, Barker, Scarer, Gern, Jagger, Swoopster and Biter—and they all make exceptional use of the

Amstrad's graphics.

The Hammer and Machine (cross-section shaped) were easily dealt with, but the Spider and Ganger (its, you've guessed—flying saucers) with their erratic movement were harder to negotiate.

Mind you these weren't that difficult. The Barker and Scarer that come next posed the first problems. Barker flies missiles ahead of them—not easy to dodge at first, but I soon got the hang of it. The Scarers were far more evil—sending their projectiles diagonally at me. Very nasty.

The Gerns did for anything—they just hurled themselves towards me at twice the usual speed.

Most of my early efforts



ended at that stage. The four lives I was given didn't last very long since when you lose one you restart at the level of nasty you were up to.

But the game's totally compelling and I kept at it—only to encounter the super-fast Jagger, who are definitely bad news. Not only are they fast, they fire at you.

After that I took the Swoopster in my stride—well, almost. These very agile birdies who launch their explosive eggs at you were a little tricky.

Then came the Biter—nasty little devils who hung back and waited for me to come near before launching their far too successful attack.

But, after thirty thirty tries,

I cleared the lot of them and found my starship—only to crash into the thing when I tried to dock. A space pilot's life is hell.

But this game is also a great deal of fun. *Galaxia*'s got the lot—good graphics, great sound and high-speed action.

It's a pity there isn't a joystick option, and that you can't select your own ship. But these minor niggles don't spoil the fun.

I can't stop playing *Galaxia*, and if I've managed to convey just a fraction of its excitement, I'm sure you'll see why.

Andrew Surinen

Game has the right spirit...

HIDDEN away in the creepy shadows at the top of the hill are a number of golden jewels. Many have searched for them but no one has lived to tell the tale. Have you the courage to go and recover where others have failed before you?

This is the challenge issued by *Shards*, Micro Power's first contribution to the Amstrad software market. It is a translation of an existing game for the BBC Micro and one of their best.

You control the star of the show, a little man looking like pac-man on two legs. His ever-munching mouth continuously snags up little fish for bonus points, as you attempt to access the various levels in your search for treasure.

You start off in the form of four screens—*Spectre's Lair*. Here to hinder you in your quest you will find the monster's ghost when you go too far.

He's a comical looking chap with an enormous frown covering most of his face. Let him catch you and you will die, wishing that brown cheese, with an enormous cheery grin. This chap follows you through every screen and at times his presence is positively painful.

You have a time limit in which to reach the face of

A better class of hangman

WHEN I had my first look at *Hangman* by Source Educational Software, and it rapidly became a great favourite with the younger members of my family.

I was therefore delighted to see it available for the Amstrad, and doubly delighted because it's even better on this machine. Too often programs are converted from one micro to another without sufficient attention to the strengths or weaknesses of the new one.

Here the sound is much stronger on the Amstrad version, which is hardly surprising in view of its vastly superior capabilities.

Hangman has appeared for

every machine available, I could imagine, but I've not seen a better version than this. It is crammed with evidence of thoughtful preparation and clear programming. The graphics are superb, yet not at all frightening—an important point with youngsters.

However my three least of the catchy little features that heralded a successful result, so they made sure they didn't hang round!

There is an extensive built-in vocabulary with six levels, and also the option to enter a single word at a time for a child to solve.

This is most useful as the parent can gauge the word

list of word to reach the child.

One of the most exciting facilities is the option to make up a word list oneself which can then be saved to cassette and reloaded at a later date.

Teachers will also find the monitor facility useful, as it displays the name of the last six children to use the program, their level and the number of successes and failures they had.

Key responses are accurate and rapid, and if my children are anything to go by, it will outlast many other educational programs as far as their enthusiasm is concerned.

Phil Taylor

jewels at the top of the screen, in order to get to the next landing. Tribble and the occasional steep jewel matched on route count for bonus points.

A jewel has the additional perk in that it makes the ghost disappear for a short while. However there are plenty of other hazards to make life unpleasant.

In order to make progress you must balance on a moving platform and leap to and from it to higher levels. There is also a set of poison-armed stragglers in your way and contact with any one of these will prove fatal.

Should you succumb on the first screen you will progress to Horrid Hall. Still pursued by the ghost, you will have to cope with not only a moving platform but also contracting floorboards. To get started you have all your disposal a large spring to propel you upwards to the first landing. There are also more poisonous stragglers on this level.

Succumbed on screen 2 and Spitter's Parlor awaits you, introducing an obvious adversary. He is something to behold but not to be touched - an additional hazard to be avoided. Fortunately he stays in one spot bouncing up and down waiting patiently for a tasty morsel. Make sure it's not you!

The infuriating aspect of the game, as with most multi-level games, is that as soon as you die you start back at the beginning of the screen no matter how far you have progressed.

I must confess, it is because of this that I haven't even seen screen 4 except in the demo mode. (You automatically go into this if you don't start the game within one minute). To be honest I've only seen screens 2 and 3 by watching the kids play - I can't get to the end of screen 1 myself!

Even so I've seen enough of the game to consider it excellent value. It is extremely addictive as there is always that incentive to "crack it this time".

The graphics are well presented but I would have thought better use could have



been made of the colours available on the Amstrad.

An outstanding feature of this package is the clever use of sound. All kinds of weird and eerie sounds make the game come to life.

My only real criticism is the use of preset keys - surely it should be standard today to allow the user to define his own keys? I think this is the main reason I never made much progress in these quick reaction games.

There is a joystick option and an option to freeze the action while you take that necessary natural break.

If Micro Power wanted to make a "spiced" break into the Amstrad market, they certainly picked the right game.

Alan Sergeant

Fancy a flutter?

I CANNOT say that I am an avid football supporter but I do take a passing interest in the results every Saturday. I have never seriously attempted the pools although I do know what is involved in making up a coupon. It was therefore with some trepidation that I looked at **Amstrad-Draw** by B.S. Mackley.

This package is a pools prediction program which attempts to forecast draws that will occur in each week's English and Scottish football league matches.

Once the main program is up and running the database on the machine must be loaded. The opening screen shows a menu of seven options.

1. Set up league position records. In order to function correctly the program must know where individual teams are in the league. So each team must be put into one of three sections - top quarter, middle half, or bottom quarter.

The program automatically brings each team's name to the screen in divisional, alphabetical order. You are then prompted for its rating. This is the tedious part of the routine and, as the instruction booklet points out, it must be

carried out every week in the early part of the season.

2. Input last Saturday's results. This must be done every week as the information is used to update the prediction database. The figures, previously defined with the next option, is displayed with the options to enter the result - 1, Home Win, 2, Away Win, 3, Draw, 4, postponement.

3. Input next Saturday's matches. These are the fixtures from which the program will predict the likely draws. The whole division is displayed, starting with the first and each team is numbered.

4. Predict next Saturday's draws. This, of course, is why you're contemplating buying the software. This option computes the probabilities of each division and asks how many of the most likely draws you require. It then displays these in descending order of probability giving the suit-



Binary tree is growing

I FOUND **World-Wise** by Bourne Educational Software to be a thoroughly worthwhile and interesting program, which will appeal strongly to parents keen to use the new machine for educational programs.

Unlike many others, this genuinely is educational because of its open-ended approach. Binary teachers and parents may have seen the Tree of Knowledge type of program, in which binary trees are used.

No, binary trees are not the ones you pick silicon chips from! They are a way of teaching the computer more about a subject, and by doing

so to reinforce ailer's own knowledge.

This program takes the binary tree idea and applies it most successfully to geography. To use an example from the World version the program can be taught about capital cities (actually there are 10 choices of subject to select from).

It knows only about two, so if you've not picked the capital of Peru or Sweden, the mine goes up. It then asks for the city you were thinking of, such as Rome. Then it asks for a question which will enable it to differentiate between say Stockholm and Rome, and which answer (yes or no)

would be correct for London.

It adds this information to its store, and can thus build up a large database with the help of the child.

There are two separate programs, one on Britain and one on the World. Each contains data on two entries in each of 10 categories. I imagine it would take a lot of information to fill the memory of the Amstrad.

Sound is used in a rather poor way and I preferred to turn it off, but the screen displays are clear. No silly colour combinations to strain the eyes here! Thoroughly recommended.

Phil Taylor

pered chance of the match being a draw.

5. Save Data. Once all your information has been updated, this facility enables you to save the new database.

6. Input individual results. This routine enables you to input results of mid-week matches without disturbing the match list you created the previous weekend.

7. Pools generator. This sets up a self-contained program which will enable you to complete your pools coupon directly from the monitor.

The screens are well laid out and the menus is easy to use. The small instruction booklet and the onscreen instructions are clear and easy to follow.

I may even take a dig in the pool myself just to give the program a real good testing. The author is quite emphatic that "Wins are not guaranteed". It is perhaps a little pricey at £9.95 but who knows - this time next week I might be able to afford one!

Alison Blacklock

Why they dig Willy

EVERYBODY seemed to know Minis 20M for the Atari when it emerged some 18 months ago. I couldn't really see what all the fuss was about.

It later spawned a glut of multi-level type games which flooded the software market and no mean competition was the top selling *Manic Miner* for the Spectrum.

Well the game is now Software Projects' first contribution to the Amstrad scene and I must admit I can now see why the game was so popular. It is fun to play, quite nice looking, and above all infinitely addictive.

Our famous hero Miner Willy, while prospecting down Sardinia, has stumbled on an ancient long forgotten mine shaft. Exploring further he finds evidence of a lost civilisation far superior to our own, where automatons dig deep into the earth's core excavating the raw materials

for their industry. And they're still at it.

Willy realises that vast wealth awaits him if he can only find the underground store. In each of 20 - yes 20! - chambers, you control the little fellow as he collects all the flashing gems. Collecting these is your only means of getting to the next chamber and it's no mean task.

While doing so you must avoid a multitude of horrors - poisonous snakes, spiders, stings, mutant telephones, man-eating toilets, bouncing chasques and many more. A single touch from any one of them means instant death.

The test is really one of strategy and timing rather than quick reactions although



on occasions it does help to be a little swift of finger to avoid the nasties.

When you first see the screen you must plan the best, and sometimes the only, route to get all the keys. Once you have mastered a screen you will normally have little difficulty in completing it every time.

There is a demo at the start which takes you through each scenario in turn and an impressive colour display it makes.

But I must admit I haven't got past screen 2 yet. I am one of the world's worst games players. I did however watch my lad get to screen 6 with not too much hassle.

There's plenty of stoibe in the movement keys as the whole of the top row are alternate for left and right.

Very impressive graphics

and sound make this a game which must have long lasting appeal to game players of all ages.

David Anderson

Counter attraction

THE Moors Challenge by Timeslip Software is a version of the popular board game Othello.

For those who have never played Othello I will describe what the game is all about. The playing board is an 8 x 8 grid onto which two players take it in turns to place circular discs. These should be positioned in such a way as to trap the opponent's pieces.

Each player has different coloured counters. The number of counters each person has determines the overall winner when either the grid has been filled or one player's counters have all been trapped.

When the game has loaded you will be asked which type of game you would like to play. There are three options. The first shows a demo of how the



game is played.

The second option allows you to play against the computer. There are five levels, from simple to advanced, which can be selected once the board has been set up.

The third option allows two players to compete. This feature is very useful if you want to challenge a friend.

The program is well presented. The instructions are not too clear, but playing the game helps you understand them more.

The game itself is excellent. Don't let me off if you consider yourself a bit of a wally. The

A question of style

Happy Writing, another from the Bourne stable aimed at three to eleven-year-olds, aims to demonstrate correct letter formation.

A "magic pencil" with the child's name written on it draws brightly coloured numbers, letters and words showing where to start the figure, which direction to take and where to end.

The child is expected to copy each figure onto paper as it is drawn on the screen. The speed at which the figure appears/pencil is adjusted to suit his stage of learning.

You can also tailor the program so that you can practise single letters, both lower and upper case, numbers, a set of similar letters - for example, c, a, d, g - your own special set and a word or set of words.

The group of three to four-year-olds I worked with had little writing experience and found it difficult to copy the letters onto their paper.

Even with the guidance of the screen they still tended to draw some letters reversed without realising it.

The style of writing the program demonstrates may not appeal to all teachers. For example, the verticals are curved at the bottom.

If however this is the style used in your school it should be useful in giving children practice in writing correctly when used in moderation.

So if you're considering computerising your classroom, Happy Writing may be of use to you.

But make sure you see it running first.

Carole Gilbert

simple game could be tackled quite easily by an eight-year-old, but the advanced level could cause problems.

Overall, a great game well worth buying.

[illegible]

Royal progress

I HATE writing outlines of adventure programs for there are two insurmountable problems.

Finally, how much do you tell the players about the game? Do you tell them not to kiss the frog until it's under the rainbow, or that the friendly dwarf is a pathological liar? Do you reveal that the pH of the Red River is more than a little less than acid rain?

Of course you don't — you'd be spoiling the adventure for the reader, wouldn't you? So not worried, none of the above refers to *Royal Guest*, from *Thematic*.

Even oblique hints like "try reading page 178 of Fraser's *The Golden Bough*" might offend the adventure reader.

The second problem is worse. You have to get sufficiently far into the adventure to let you feel the force of it — and, of course, the program-makers aren't going to make it easy for you, are they?

What the tireless reviewer prays for is a game that starts simply, has enough "classical" themes to give him something to latch onto and then one or two real problems to let him put the Bayes of the game

Fortunately for me, Royal Quest provides all these. You start on an East-West path, with a palace in the West. I suggest you visit the palace first in order to find out what the game is about (another stolen crown) and to receive your sword.

From there on it's just along the path, with diversions to visit a cemetery and overgrown orchard – picking up one or two things – then via a sleeping dwarf into the forest.

Oh, her are adventures with out a sword! Well, this year isn't too bad, and you're soon on the

edge of the lake where the
currents really change.

Incidentally, you need "GO BOAT" here – the only questionable bit of vocabulary I've come across. Normally the commands you enter are quite sensible – and you can abbreviate them to three letters, with single letters for M, H, E and so on.

All in all the game is standard, enjoyable adventure fare, with enough tricks and puzzles to maintain interest, plus one or two nice touches. I particularly liked the book packed full of information about vitamins, shoppers and the like.

And the game was well presented, with text appearing on a parchment scroll, making use of windows to separate the race descriptions from the demands and responses.

It was a pity the effect was spoiled by a plethora of misspellings that gave the game a totally undesired antiquarian feel at times.

That mean apart, *Royal Quest* is enjoyable and entertaining. Perhaps not quite a classic, but with enough interest and difficulty to make me glad of the facility for saving unfinished games.

1. *Journal of the American Medical Association*, 2000; 283: 2686-2692.

Borrow a Biggles

WHEN it comes to flying I've played as useful as a chocolate fireguard, but I was determined to try out Airbus's jet trainer simulator *Flight Path*. **799**

You take off from an airfield surrounded by high mountains, and having climbed safely over them, prepare for a landing in the valley below.

There are six levels of play ranging from "first solo" to "Test pilot". I curbed my natural desire to try "stunt pilot" and selected the first level.

The screen consists of a split display in Mode C with the view from the cockpit above and instruments below.

The hotel policy's main problem, then, takes off passengers.



Press T and you will start to list at 20 knots (kays 7,4,3) control speed by +20, +30, -10 and -20 knots respectively. I put the flaps down as requested and tried to line myself up with the runway.

I found this very tricky to start with as the key response was slow and regularly I had overshoot the target even after releasing it is joyful, or the left and right cursor keys control lateral movement. You have to get the center white line dead straight on also.

"Increase speed to 180 knots but not more than 200 because the flaps will be damaged", you are advised. I must have caused thousands of pounds worth of damage to these flaps before I finally got it right.

Roll the joystick back to
use the down arrow key) and
the jet will take off. Sure thing!
As you climb you have to keep
your finger on the throttle to
maintain air speed.

Once above 300 feet you can't retreat the undercarriage (A) and the legs (F) — and this is where the trouble starts.

Apparently when you raise the productivity you are not

craft the speed will increase by 5 knots as it is to be cancelled at this phase not to exceed the minimum 100-knots turn.

We climbed higher and the scenery changed to mountain tops as we leveled out at 8,000 feet. With the mountains approaching a red ground warning flashed but dissipated when we were safely above the hazard. When the light went out we were clear of the mountains and ready for landing.

This is a nice piece of software for the flying enthusiast and games player alike and should be very popular.

The graphics are well done and the sound is adequate. I was a little dubious about the slow key response but I don't fly enough to know if this is standard on flight simulators. After all you can't turn a 737 around in a minute.

If you want a challenge that's going to have you kicking the cat away 5 minutes, then look no further.

[illegible]

Be a Big Mal

FOOTBALL Manager by Addictive has to be one of the best strategy games available for home computers.

Already a local center for a number of other machines I found myself and my friends became totally enthralled with the game, forgoing meals to have just one more match.

As the manager of a football club you have to get from league division four to league division one in as few seasons as possible.

The first screen asks your name and allows you to select your team from the whole league. There seems to be little advantage in picking the well-known clubs as they are all dependent upon the skill of their managers — *same*.

The team you intend to train is then displayed, together with six pieces of information on each player – playing position, number, skill rating, energy rating, saleable value and

REVIEWED THIS MONTH	
Administration	100 (34%)
Flight Risk: 700	40 (13%)
Financial Manager	10 (3%)
Formal or World's Best	10 (3%)
Finance	10 (3%)
Quality	10 (3%)
Human Resources	10 (3%)
Marketing	10 (3%)
Management	10 (3%)
Operations	10 (3%)
Product Development	10 (3%)
Research & Development	10 (3%)
Sales	10 (3%)
Training	10 (3%)
Other	10 (3%)

status (either picked or injured).

From this screen there are a number of other options affecting the game. You can get a display of your performance, beg for a loan from the bank and even pay off loans.

You can also affect the game settings if you wish. You can change your skill level, change the team and player names, save a game or restart a saved game.

Once you have done all the housekeeping from the first menu you have an announcement of your next fixture, which might be in either League or Cup competition.

Following this there is a readout of the comparison of the teams giving ratings for Energy, Morale, Defense, Mid-Field and Attack. This is worth careful study as it is here that you can achieve the most valuable changes by modifying your team slightly.

Presuming that you have enough players on your books you can arrange to strengthen your team to meet the opposition's rating. It is said that one of the most important

skills of management is the testing of players who then build up their strength and morale. It certainly seems to matter in this game.

You have chances to sell your own players and bid for others. Unsuccessful bids mean an increase in the selling price so be warned.

I doubt that this game will ever be bettered. It is at a claim to be and is very addictive to boot.

Dave Carlos

Time...

TIMEMAN One, from Broom Educational Software, is designed to help children aged 4 to 9 to tell the time. The first stage of the program, for the younger child, introduces the hour hand only, the second stage the minute hand, and the third and final stage combines the two.

Having taught how to tell the time, the program then provides three similar stages to enable children to set the clock. Time is expressed throughout in minutes past the hour in line with digital clocks

and watches.

The graphics are appealing. There is a large clock face, and a man placed half way up a ladder. The program asks: "What time is it?" If the child answers correctly by pressing the appropriate number of keys the man climbs two steps up the ladder.

When the answer is wrong he goes down a step, the child is told what is wrong and given another go. When the man reaches the top he dances a jig and plants a flag.

Setting the clock is again in three stages — first setting the hour hand and its pressing the 11 key to rotate the hand, and then the minute hand by pressing the M key. This can be done in one or five minute intervals as required. Finally both hands can be adjusted together.

This is a very handy package which the children will enjoy working through.

It covers a wide range of abilities — suitable for 4 to 9-year-olds who can practice telling the time by the hour hand only and 7 to 9-year-olds who can use both hour and

minute hands. Definitely worth having in the classroom.

Carole Sillescu

and time again

TIMEMAN Two follows on from Timeman One but can equally well be used on an independent basis. It is designed to give children aged 4 to 10 a greater understanding of time, including the 24 hour clock.

It covers minutes to the hour, the quarter and half hour as well as the 24 hour clock. As in Timeman One, the children are required to tell the time and to set the clock. This can be in one or five minute intervals to suit each child.

As before, a little man climbs up a ladder each time the child gives the correct answer.

Again a very useful program giving children the practice they need in telling the time, but in a way that is not boring.

Carole Sillescu

Superb scenery at World's End

AS soon as I saw the opening screen of **Forest at the World's End**, the latest adventure from Interceptor Software, I knew I was in for a real treat.

To say that the title page shows a castle on top of a hill, surrounded by trees is accurate, but it totally fails to convey the quality of graphics involved.

Quite simply they're superb — better than anything I've seen on other systems.

As the title finishes loading you find yourself in the Great Valley before a forest — again beautifully shown, conveying that ethereal air so vital for an adventure.

You won't get a picture for every location you visit, but still there are an enormous number packed into the game. I could continue raving about the graphics, but, after all, it's supposed to be an adventure game not a picture book, so

how good is it?

Well, it's not bad at all. You'll find plenty to intrigue and bewilder you as, in your role as the mightiest of ancient warriors, you try to rescue Princess Mara from the clutches of Zari, the evil wizard.

However I get the impression that more effort has gone into the graphics than the plot, which is rather predictable.

All that is, except the combat. Having collected your bow and arrows you are ready to repulse attackers that — go south first.

But what a boring business it is. No skill, just totally random results as you loose your arrows. I find this aspect of adventures a definite minus, though I admit that I'm the type who prefers chess to Monopoly.

In mitigation, there is the Rune feature, careful use of which can make the random mortality less final.

By the way, to fire your arrows enter: **ALL, ELF, WOLF, GOW**. It's the bow you use, not the arrow!

A nice feature of the game is the command syntax. As shown above, you can get away with far more than the normal two word commands of other adventures.

The game is thoroughly enjoyable, even if the objects do tend to get presented to you on a plate.

The first major obstacle is the chasm, which isn't a problem: provided you've thoroughly looked at the other locations on this side of it.

Then there's a journey over the plain to the bank of a river — don't cross, though, until you've paid the old well's debt.

After one or two more incidents you come to a precipice. No problem provided you remember there's a dragon watching you.

Then you'll want to pay the

witch a visit before you go through the Mar's gateway...

As you can see, there's plenty of action, and your interest level is maintained throughout. All right, there may not be much innovation, but that's not Mills and Boon's strong point either, and they still sell.

And, of course, the graphics take it well out of the ordinary class of adventure games — although I would have liked them to be an integral part of the plot rather than merely decorative.

Still, **Forest at the World's End** is an extremely enjoyable, entertaining game and deserves to be successful.

After this good start, I look forward to more graphic adventures from Interceptor. They're going to be even better.

Justin Marston

Meet Smiley and beat the galloping Grumpies in
ROLAND WADDILOVE's version of an old favourite

Introducing Smiley...

In this colourful and competitive game you have to guide Smiley round a maze picking up coloured buttons that are scattered about. The buttons belong to the Grumpies, who take exception to you plucking them from under their nose and chase you round the maze, trying to catch you.

If you collect all the buttons in the maze it is redrawn and the game starts all over again, except that you and the Grumpies move a little faster.

Yes, it's an Amstrad version of the old favourite — no collection is complete without it! There is a bonus which decreases with time, three lives and a keyboard or joystick option.

The program is fully structured with no GOTOs to confuse you. It is controlled by calling the various subroutines from lines 40 to 210 to print the instructions, draw the maze and move the characters.

Each subroutine has been given a role in a REM statement at the start of it so you know where to look if it does not work first time (it's nearly impossible to type in a program without making at least one mistake).



SUBROUTINES

Timer/interrupt	I haven't worked out how to set the clock yet so I have used Timer 1 to generate an interrupt and increment my own clock.
Bonus interrupt	Timer 0 is the bonus counter, counting down to zero in 100 seconds. The bonus is printed in window 1 so as not to upset the print positions and colours of the other characters.
Initialise	Draws the arrays, sets the colours and the high score. 4% is the "Y" key.
Draw the maze	Reads the data statements and adds 82 to the Ascl code of each character before printing.

Start positions

Sets the start positions of the Grumpies and Smiley. Sets the bonus and the dots left. Reduces delays. Sets score and delays to initial values. Prints score and three Smileys.

Set up

Move man

If it is not time then return. Sets the next time. Finds the new coordinates, and sets the flag (ok) if they are the same as one of the Grumpies. Prints at the new position, erases the old one. Increments score if a button has been picked up.

Move ghosts

Calls ghost to move the Grumpies if it is time.

Ghost

Looks where Smiley is and moves towards him if a wall is not in the way. Checks if he has been blighted. Replaces button if necessary.

**Caught**

Makes a rude noise and flashes several characters. Decreases lives. The large title is drawn by printing it at the bottom in black, looking at the points and plotting it at the top of the screen. Prints the instructions, sets either keyboard (default setting) or joystick mode.

Next screen

Prints screen completed in transparent mode. Adds the bonus to the score and increments the screen.

Game over

Cleares the screen by drawing black lines toward the centre. Prints scores. Asks if you want to play again.

`rl`
`bonus%`
`maxsc$000`

VARIABLES

Timer.
 Bonus.
 The max.

`ghost%(2,1)`
`grime%(2)`

`gldelay%(2)`

`rl%`

`rl%`

`rl%`

`ok`

`maxsc$,many%`

`rl%,y%`

`score%`

`dots%`

`lives`

`screen`

`rl%,b%,c%,d%`

`minlives%`

`mdelay%`

Grumples' positions.

Time when the Grumples next have to move.

Delay for the Grumples.

Used in loops.

High score.

General variable.

A flag to show whether caught.

Smiley's coordinates.

Temporary coordinates of Grumples or Smiley.

Scores.

How many dots left.

Number of lives left.

Number of screen.

Values used by INKEY()

The time Smiley can next move.

The delay for Smiley.

```

10 REM *** SILENT ***
20 REM My R.A. Assistant
30 :
40 GOTO 1:GOTO 1:END
50 GOTO 8:GOTO 300
60 WHILE <=41
70 GOTO 100
80 WHILE 1line
90 GOTO 400-GOTO 010
100 EXEY 3,1:GOTO 200
110 EXEY 50:GOTO 270
120 WHILE ok AND detd
130 GOTO 1000:GOTO 1000
140 END
150 10-REMMAIN
160 IF ok THEN GOTO 1000 ELSE GOTO 1700
170 END
180 GOTO 1720
190 END
200 GOTO 1
210 WHILE INKEY<>"**END"
220 END
230 :
240 10-1+1:RETURN:REM time interval
250 :
260 REM #1 time interval =
270 IF bound THEN bound:=bound+1:1:1
280 01,1,1:PR#10 "R,nowe&detd&n
290 RETURN
300 :
310 REM == Initiation ==
320 DIM macr(20),phon(11),gnd(11),phaly(11)
330 RESTORE 700
340 FOR i=0 TO 11
350 READ Jcr(i),JL,JL
360 NEXT
370 GROUND 0
380 st=&1:1:1:0
390 RETURN
400 :
410 REM == Draw the card ==
420 LOCATE 1,1
430 RESTORE 100
440 FOR j=1 TO 25
450 READ macr(j)
460 FOR j=1 TO 20
470 st=&+macr(j)/macr(11),j,1:1
480 IF <0:147 THEN FOR screen ELSE PG
490 screen=
500 PRINT CHR$(1);
510 NEXT
520 NEXT
530 RETURN

```

[illegible][illegible]

WELCOME to Bits and Bytes, the first in a series of articles in which we hope to take the mystery out of understanding the fundamentals of the Amstrad's workings.

All too often even competent Basic programmers tend to shy off such topics as binary coding, hexadecimal and assembly language because it seems too "mathematical".

This is a great pity, because the CPC464 is so constructed that a little knowledge in these fields allows you to take full advantage of its advanced facilities.

The mathematical aspects of the subject aren't at all deep - certainly anyone who can follow Basic should be able to cope with this series.

If you feel that despite our best efforts we still haven't explained something fully enough, please write in and tell us - we'll try to rectify the situation in later articles.

First we are going to look at binary code - a way of handling numbers essential to our understanding of what goes on inside a computer.

Binary is just a way of coding numbers in a way particularly suitable for computers. It's actually quite simple. What often confuses beginners is the fact that the binary system codes numbers in a way that can look extremely like the way we normally code numbers.

For example, if you were presented with a number 100, you would probably decode it in your normal way and say it was "one hundred".

That, however, is just one way of interpreting it. If you decided to decode it as a binary number, you would interpret 100 in a completely different way and say it meant the number "four." (We'll asked exactly how you arrived at that conclusion for the moment.)

This is what often causes problems - people are so used to dealing with their numbers in the normal way that 100 is always "one hundred" to them, and they can't make the shift necessary to decode it in binary as "four".

Actually it is rather ambiguous. Presented with 100, do you interpret

Let's lift the hex on hexadecimal

it as "one hundred" or "four"? Our rule will be, if you mean our usual way of dealing with numbers (*like Americans, tens and units you learn at school* - or put it more formally, the decimal system) you write the number in the normal way.

If you wish the number to be decoded as a binary number you put the symbol % in front of it - 100 means "one hundred" while %100 means "four".

So far so good. We now have a number (4) to work on that we have to

carry out unnecessary amounts of change, and I for one don't like doing that.

Sometimes, however, with our present coinage system we have to use the same coin twice to obtain certain sums. You cannot, for instance, make up the sum of 4p without doubling up on coins. To avoid repeating coins we would have to invent a 4p coin!

Let's do that; in fact, let's invent a coinage system where you never have to use the same coin twice.

First of all we would need a 1p coin and, of course, a 2p coin, because we cannot use 1p + 1p for 2p - it breaks the rule!

Now 3p can be made up of 1p + 2p, but for 4p we'll have to invent a 4p coin.

Equipped with that we can make 5p (4p + 1p), 6p (4p + 2p), and 7p (4p + 2p + 1p). In obtaining 7p we used all our available coins, so now we have to invent an 8p coin. If you work it out (and I suggest you have a go) you will find that with the coins you have at your disposal (8p, 4p, 2p, 1p) you can make any sum up to 15p. Then you would have to invent a new coin, 16p.

Notice how the coins we have created have doubled in value: 1p, 2p, 4p, 8p, 16p. No prices for guessing what the next one is.

Let's summarise our results in a table (Figure 1). Here I have used the columns to show the coins available and the rows to show how the various

By MIKE BIBBY

decode the number in a special way as a binary number.

However, before you decode you need a rule for decoding - as how do you get the number "Four" from %100? What's the rule?

Let's take a detour for the moment, and think about the coins we use every day. Our currency consists of these coins:

50p, 20p, 10p, 5p, 2p and 1p (ignoring the half-penny). We can combine them to give any sum we wish. For example:

75p is 50p + 20p + 5p
or 50p + 10p + 10p + 5p
and so on. We are all familiar with this - often we use multiples of coins to make up a sum. For example, 5p can be 2p + 2p + 1p.

Using the same coin twice, though, often means that we end up

totals are made up. A 1 in a particular column means that we use that column's coin, and 0 means that we don't use it. Look at the row for 5p. It has 101 on it. According to our rule, this means we pick out the coins 4p and 1p (and NOT 2p) to make up the 5p total.

$$\begin{array}{r} 4p \quad 2p \quad 1p \\ \% \quad 1 \quad 0 \quad 1 \\ + \quad 4p \quad + \quad 1p = 5p \end{array}$$

Now let's get back to computers by dropping all this talk about coins and redraw Figure 1 to show the same information, but without referring to money — just numbers. Figure 1 is the new table.

As you can see, there is little change, and we can use this table to encode numbers in general, not just coins. We call this method of encoding the binary system.

Remember, to show that we mean a binary number we precede it with %. So if you see, for example, %101 means:

$$\begin{array}{r} 4 \quad 2 \quad 1 \\ \% \quad 1 \quad 0 \quad 1 \\ + \quad 4 \quad + \quad 1 = 5 \end{array}$$

that is we add together the values of

the columns containing 1. Look at row 5 of the table to check it. Similarly, %101 would mean 13 in the binary system since

$$\begin{array}{r} 8 \quad 4 \quad 2 \quad 1 \\ \% \quad 1 \quad 1 \quad 0 \quad 1 \\ + \quad 8 + 4 \quad + \quad 1 = 13 \end{array}$$

By now you should be able to work out for yourself why %100 represents four. From the table, or by using the addition method I've just illustrated, see if you can decode the binary values of the following binary numbers:

%1001
% 101
% 11
%1101
% 111

You can use the program accompanying this article to check your results. You've probably noticed by now that in the binary system you only use two symbols, 0 and 1, to encode numbers — hence binary, for two as in bicycle.

You can encode any number that you want in binary — just use more columns (or "bits" as we say in computer jargon), remembering that

each new bit is worth double the preceding bit.

However it does get terribly cumbersome. For example, 100 (tenary) encoded in binary is %1100100 since

$$\begin{array}{r} 64 \quad 32 \quad 16 \quad 8 \quad 4 \quad 2 \quad 1 \\ \% \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \\ + \quad 64 + 32 = 96 \end{array}$$

It is much easier to handle the number in our normal system. To a computer this presents no problem, and the fact that binary only uses two symbols is a bonus because you can represent numbers with a sequence of "switches".

Switches are what we call "two state" — they're either ON or OFF. If we have a sequence of four switches together we can encode numbers by having them either ON or OFF. We could use ON to mean a 1, and OFF to mean a 0 in a particular column:

$$\begin{array}{r} 8 \quad 4 \quad 2 \quad 1 \\ \text{ON OFF ON ON} \\ + \% \quad 1 \quad 0 \quad 1 \quad 1 = 11 \end{array}$$

Each of these "switches" represents a bit, and a computer memory is full of bits. The Z80, which is the microprocessor at the heart of the

COINS				
	4p	2p	1p	
1p			1	
2p			1	0
3p		1	1	
4p	1	0	0	
5p	1	0	1	
6p	1	1	0	
7p	1	1	1	
8p	0	0	0	0
9p	0	0	0	1
10p	0	0	1	0
11p	0	0	1	1
12p	1	1	0	0
13p	1	1	0	1
14p	1	1	1	0
15p	1	1	1	1

Figure 1

Binary Value	Column 8	4	2	1	Binary Value
1				1	%1
2			1	0	%10
3			1	1	%11
4		1	0	0	%100
5		1	0	1	%101
6		1	1	0	%110
7		1	1	1	%111
8	1	0	0	0	%1000
9	1	0	0	1	%1001
10	1	0	1	0	%1010
11	1	0	1	1	%1011
12	1	1	0	0	%1100
13	1	1	0	1	%1101
14	1	1	1	0	%1110
15	1	1	1	1	%1111

Figure 2

Amstrad system, deals with 768,432 of them.

To make things simpler the 255 handles the bits in groups of eight bits at a time — the group of eight being called a byte.

With this type of organisation the largest number you can store in a byte is 255 since:

```
128 64 32 16 8 4 2 1
% 1 1 1 1 1 1 1
→ 128+64+32+16+8+
  4+2+1=255
```

Of course the computer can handle larger numbers (and not just whole numbers) but to do so it must use more than one byte.

Converting a byte from binary to denary is fairly straightforward. Simply write it down under the appropriate column for bit values and add together the value of all the columns in which a 1 occurs. For example, given %10010101 you translate as follows:

```
128 64 32 16 8 4 2 1
% 1 0 0 1 0 1 0 1
→ 128+16+4+1=145
```

Going from denary to binary is not at all difficult, but is rather hard to put into words. You do it by subtracting from the number you want to encode the value of each column in turn, starting with the highest (i.e. 128, 64, 32) and so on.

If you can't subtract a particular column value you put a 1 in that column and continue to subtract the

next lower column value from the remainder.

If you cannot manage the subtraction you put a 0 in that column and try to repeat the subtraction with the next lower column number.

So, starting with the highest column number (128 in our case), you:

WHILE you have done all eight columns:

1. Attempt to subtract the relevant column number (highest first).
2. If you succeed then put a 1 in that column number and continue to subtract other columns from the remainder. If you fail, put a 0 in that column.

WEND

(Figure 8) should make it clearer.

In practice, when faced with encoding a number from denary to binary I tend to do it in my head, seeing which column values will add together to make the sum required, starting with the highest first.

For example, if I want to encode 161 in binary I would say, "Well, I can use 128, so that leaves me 33 to find. 33 can be made up of 32 and 1 so that does it: 128+32+1=161. So I encode it as:

```
128 64 32 16 8 4 2 1
% 1 0 1 0 0 0 0 1
= %10100001
```

After a while you'll find this way quite simple.

To finish off, I'll leave you with a

program to print out the binary value of a number between 0 and 255 (i.e. that can be stored in one byte). Try it with various values and see if you can accept the results.

The program itself uses one or two ideas, such as AND, that may not be too familiar to you as yet.

```
10 REM *****
20 REM *** AMSTRAD CROMAT ***
30 REM *****
40 MODE 1
50 WHILE -1
60 number = -1
70 WHILE number > 255 OR number < 0
80 LOCATE 1,1: PRINT STRING$(24," ")
90 LOCATE 1,1
100 INPUT "Number":number
110 WEND
110 PRINT:PRINT "Is binary 'number'"
120 J=1
130 PRINT (J=1) number,0
140 LOCATE 1,2: PRINT "this works out as"
150 FOR I = 7 TO 0 STEP -1
160 LOCATE 30-(4+I),J: PRINT USING "###"
  "12"
170 LOCATE 30-(4+I),J: PRINT USING "###"
  "(1 2) AND number / 2"
180 PRINT:PRINT
190 NEXT I
200 WEND
```

Worry not, "Bits and Bytes" will cover them. Watch this space...

```
140
-128      128 goes - set to 1
      21      64.32 can't go - set to 0
-16      16 goes - set it to 1
      5      8 can't go - set to 0
      4      4 goes - set to 1
      1      2 can't go - set to 0
-1      1 goes - set to 1
      0
```

128	64	32	16	8	4	2	1
1							
	0	0					
			1				
				0			
					1		
						0	
							1
1	0	0	1	0	1	0	1

Figure 1W

Program

```

700 IF INSTR("y",answer)=0 THEN G
  O 5070 00
710 END
720 REM *****
730 REM *****
740 REM *****
750 REM *****
760 REM *****
770 REM *****
780 FOR I=1 TO 10:LOCATE 4,2:PRINT "Welcome
  to ";LOCATE 5,4:FOR S=PRINT "LETTER
  LITTON"
790 LOCATE 1,8:PRINT "A letter or an
  other"
800 LOCATE 1,8:PRINT "what will you
  say"
810 LOCATE 1,10:PRINT "on the screen"
820 LOCATE 5,10:PRINT "press the right
  key"
830 LOCATE 1,10:PRINT "and a small star
  will"
840 LOCATE 1,10:PRINT "appear. Keep it
  a key"
850 LOCATE 1,10:PRINT "pressed and be

```

```

will"
860 LOCATE 1,20:PRINT "collect the let
  ter."
870 LOCATE 4,25:PRINT "Press (SPACE)"
880 IF INSTR("y",answer)=0 THEN 800
890 GOTO 5070
900 LOCATE 1,1:FOR S=PRINT "Don't fo
  get to use"
910 LOCATE 1,3:FOR S=PRINT "the wit
  1 key when"
920 LOCATE 1,5:FOR S=PRINT "success
  ty."
930 LOCATE 1,7
940 FOR S=PRINT "You can decide the"
950 LOCATE 1,9:FOR S=PRINT "game long
  th...."
960 LOCATE 1,13:FOR S=PRINT "at 30 se
  conds"
970 LOCATE 1,15:FOR S=PRINT "at 1 min
  ute"
980 LOCATE 1,17:FOR S=PRINT "at 2 min
  utes"
990 LOCATE 1,20:FOR S=PRINT "press a,
  b or c"

```

```

1000 WHILE INSTR("y", answer)
1010 LOCATE 1,20:PRINT
1020 INPUT letters:IF INSTR("abcd",
  letters)=0 GOTO 1000
1030 IF letters="a" OR letters="b" OR
  letters="c" THEN GOTO 5070
1040 IF letters="d" THEN GOTO 5070
1050 IF letters="a" THEN GOTO 5070
1060 IF letters="b" THEN GOTO 5070
1070 IF letters="c" THEN GOTO 5070
1080 GOTO 5070
1090 RETURN

```



Give your fingers a rest....

All the listings from this month's issue are available on cassette. See our special offer on Page 55.

AMSTRAD CPC 464 SOFTWARE



ADDICTIVE.....

AMSTRAD CPC 464 SOFTWARE



PERPLEXING.....

AMSTRAD CPC 464 SOFTWARE



CHALLENGING..

WHAT MORE DO YOU WANT?

All programs DS-95 inc. p&p. from

TIMESLIP
SOFTWARE

SECRETBURN WORKSHOPS
THE OLD PRIMARY SCHOOL, MAIN STREET
STONERBURN, WEST SCOTLAND, SCOTLAND EH47 8AF

Orders Enquiries Welcome



Turn Grumpies into Smilies in ROLAND WADDILOVE's guessing game

FIVE Grumpies are standing behind a wall. Can you find out what colours they are and their order? You can choose how many different colours the Grumpies may be and you can have up to 10 guesses.

Pressing one of the number keys places a coloured Smiley and DEL will delete the last one. When you have entered a row of five Smileys your guess is marked. A black peg is given for the right colour in the right place, and a white peg for the right colour in the wrong place.

Many of you will have guessed the game it is based on. If you haven't it is very easy to learn and great fun to play, but frustrating when you can't work out the code!

The program is fully structured with no confusing GOTOs. Lines 30 to 50 run the program, calling the various subroutines when necessary. Each of the subroutines has been labelled with a REM statement and they are separated by lines with a single colon.

[illegible]

THE Amstrad CPC664 is capable of supporting all manner of devices which we can attach to it from the outside. But before we can do this we have to know a little bit about what goes on inside the computer.

At the back of the machine are three connectors to which we can attach devices. These are labelled "User Ports", "Floppy Disc" and "Printer". Although these names indicate what you CAN put in it, you are by no means limited to just those devices.

Let's take a look at each connector in turn and see what potential they have.

The User Port is designed to be connected to a pair of joysticks and is scanned along with the keyboard. That means you can read the state of any individual input by using an INKEY command. The codes returned are printed on Page 16 of Appendix III of the User Instructions book.

The way this is put together inside the machine means it can only be used as an input, unlike many other computers' user ports which can be used either as an input or an output.

Nevertheless, many interesting devices may be connected to this port such as pressure mats for burglar alarms or an analogue to digital converter.

The printer port allows you to connect to a standard centronics parallel printer interface – the type used by most printers. However you could also use it as a digital output for many devices.

Normally a printer has an 8 bit output but in the case of the Amstrad it only provides a 7 bit output as the connector marked D7 (pin 9 of the 34-way edge connector) is connected to earth.

While this will not affect normal printing of letters and numbers, some printers use the output to obtain special effects. The Epson FBDD for example uses it to select an alternative italic font.

It is also used in the graphics mode to specify the number of points to print and the state of the eighth needle. This can be circumvented by programming but it does make it harder than necessary.

In fact, certain printers which cannot be made to alter their line spacings will not be able to produce graphics dumps.

The printer latch is located at address 1FBDD and can be accessed by using the OUT command. This can be used to switch all manner of devices on and off such as lights and electric hatches.

However as the voltage coming out of the socket is only 5 volts this will have to be buffered up by extra components before it can switch anything useful. It can also be used to control other electronic devices directly like a small robot or a digital to analogue converter.

Finally we come to the connector marked "Floppy Disc". This is potentially far more useful than any of the other connectors as it allows access to almost the entire bus of the microprocessor.

In addition it allows us access to the light pen input of the 6945 CRT controller chip, which controls the generation of the video display.

By feeding the signal from a light pen into the chip it will register the position on the screen where the pen is pointing. This can be used for drawing or painting at menus and is an easy way to interact with your computer.

Inside the chip are certain registers which may be altered to give almost instantaneous sideways scrolling.

This is used to great effect in the types of games where you are given a side view of your rocket and have to guide it over mountains and around tunnels while blasting away – and not being blasted in return.

As we have access to the whole bus of the microprocessor we can also graft extra devices into its memory map. The 280 microprocessor inside the Amstrad computer is one of the few microprocessors to have a separate input/output address space.

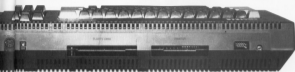
Normally in a computer system memory address spaces are used for input/output devices. This means we can have a maximum amount of RAM and ROM to store our programs and still have all the input and output we want.

It also means that the 280 provides special instructions to input and output data and does not rely on the conventional memory access instructions.

The 280 microprocessor was designed to have 156 bytes of

Amstrad's to the big

*First of a series by MIKE COOK
on expanding the Amstrad's
horizons with add-ons*



s ports - back door g outside world

input/output space but the designers of the Amstrad computer seem to have improved upon that.

Certain input/output instructions place one of the internal registers (the B register) on the higher 8 bits of the address bus, which in effect gives us 64k of input/output space. However the designers then seem to have squandered this improvement by making each device take up 32k bytes of address space by incompletely decoding the address bus.

The result is that we have apparently gained nothing and lost the use of some important input/output instructions such as those which can transfer a block of memory directly to an output port.

This strange behaviour may have a rational explanation. It may serve to simplify the circuit, thus reducing the component cost by almost £1. What a price to pay!

When we are considering designing any add-ons in the address space we have to make sure that we do not conflict with any devices which use this space inside the computer.

The manual which accompanies the computer makes a passing reference to some of the locations used. Table 1 contains a fuller list.

Note that we are not restricted to decoding the higher order address lines so we can have a large amount of space for our devices. But

remember we cannot use these powerful input/output instructions lurking in the Z80 as it would upset the inside of the machine.

In essence, designing an input/output port is simple. We monitor all the address lines, until the combination we have chosen for our device appears. If at this time the I/O line tells us this is an input/output operation all we have to do is to place data onto the data bus or record the state of the bus depending upon what the read/write line is telling us to do.

There are many large scale integrated circuits designed to do

this. One low cost one is the D8255. This is useful to examine as it is the same type that is used inside the Amstrad to look after the keyboard, user port input and sound chip.

Figure 1 shows the internal configuration of this device. You will see that it essentially consists of three parts, which means we have three separate 8-bit input/output address locations. The fourth location is a control register which effectively tells the three parts how to behave.

It is a complex device and can be configured in a number of different ways. Each one is called a mode of

Address	Use
8F4XX	Port A of the internal D8255
8F5XX	Port B of the internal D8255
8F6XX	Port C of the internal D8255
8F7XX	Control register of the internal D8255
8F8XX to 8F9XX	Page expansion bus *
8F1XX	Printer output latch
8D1XX	Expansion ROM
8B0XX	6845 CRT Controller address latch
8B0XX	6845 CRT Controller data latch

* Indicates future expansion. This device is not fitted to the standard computer.
X indicates that any two digits may be used and the device will still respond. The digits used have no effect at all so they might as well be 00.

Table 1: The internal input/output address map

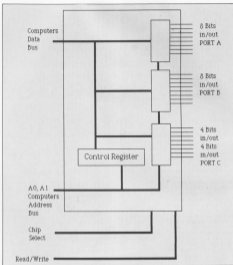


Figure 1: 6825B input/output chip

operation. Figure 11 shows how to write to the control register to obtain the various modes.

In Mode 0, ports A, and B can be defined as simple inputs or outputs. Port C is split up into two 4-bit halves, each half of which can be an input or an output.

In this simple mode of operation you can read the state of the logic bits on the inputs or change the state of the logic bits by writing to the

outputs.

Mode 1 has ports A and B assignable as inputs or outputs. But this time port C acts as handshaking lines to the two ports.

A handshaking line is a line that tells us something about the data on the ports. It helps transfer the data by indicating when fresh data is available or when a device is not ready to accept data.

Handshaking is used mainly when

two systems are talking together, such as the microprocessor in the computer "talking" with the microprocessor in a printer, XY plotter, storage system or even another computer.

Mode 2 is available only on port A. This makes the port act as a bi-directional data port, which means data can be sent and received from the one port.

In this mode port C provides the

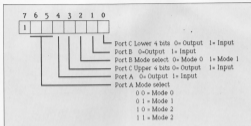


Figure 8: The control register

handshaking. This is useful for connecting the data buses of two computers together. Note that when port A is in mode 2 then port B may be in Mode 0 or Mode 1.

Finally there is a method of setting or resetting any individual bit in port C. To do this you write to the control port with a logic one in the most significant bit and a zero in the least significant bit if you want to reset a port C bit, or a one if you want to set it.

The number of the bit you want to alter is placed in bits 1, 2 and 3. Figure 9 illustrates this operation.

As I said, there is one of these devices lurking inside the Amstrad computer. I think it is safe to assume that it is operating in Mode 0.

As a matter of interest, the **BUSY** signal from the printer port which tells the computer to hold up any new data until it prints the last one is fed into this port. If you want to find it, it goes to bit 6 on port B.

You could use this in your programs to see if the printer is online before you send it any data.

The Amstrad is a fairly new computer with not many additions available for it yet. However this will change.

You may even decide to make some yourself, in which case "watch this space" for complete projects and hints on designing your own.

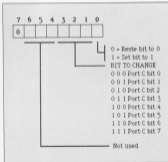


Figure 9: Changing bits on Port C

I KNEW there would be trouble when she saw it in the middle of the dining room table.

"What on earth have you bought now?" she asked with an air of complete bewilderment reminiscent of the occasion I turned up with a moped after popping out to buy a paper.

"It's an Amstrad CPC464", said I with forced nonchalance.

"What do you want with another music centre?" she asked. "It's not a music centre love, it's a computer", I replied steadily.

"Not another one?" she demanded, glancing at the console snugly housing my BBC Micro. "What are you doing - collecting them? Mind you the screen's nicer than the other".

I must admit though, the canary yellow text liberally sprinkled across a bright blue background did brighten the place up a bit. It added a beautiful backdrop to the condiment set and the cracked vase from granny that we don't throw away for fear of reprisals from her gang of senile delinquents.

I don't collect computers, but they have become both my work and play. I am no expert programmer but I have limited experience of two other machines.

I bought a Tk2480 three years ago when I was a copper. Well, someone had to. Seriously, it was cheap enough to have a play with and I could throw it away if I didn't take to it. But of course, I did - look, line and sinker.

Within a fortnight though I was fed up with running out of memory - (and the micro was a bit short of RAM as well) - *Zed*. We needed to expand. I sold it to the newspaper lad and bought, in my opinion, the most economical keyboard computer around at the time - the Dragon 32k! Just think of all that extra space.

I got a lot of pleasure out of the Dragon and became reasonably proficient at Basic. Since then I've

progressed onto the BBC Micro and finally the Amstrad. Which is where we came in.

When I arrived home with it I was bubbling with excitement. A £45



micro, a cassette recorder and a colour monitor all in one package was an offer I just could not refuse.

I unpacked the hardware with trembling fingers. At last, with protective polystyrene scattered to the four corners of the room and the cat making a bed out of the discarded plastic bags, it was there in all its glory - glorious!

Why, oh why, is the inclusion of a 3 amp plug such a daunting prospect to the manufacturers of many electrical

components? Where do you get one from at 6.30pm on a Friday?

Of course I should have anticipated this omission (I do at Christmas with battery-operated things for the kids, but I made light of it, literally. There is now no plug on the sideboard lamp.

I'm not behind the door when it comes to matters electrical. I did, however, make a slight - what they would call in the trade - technical faux pas. Before attaching the wire to the terminals, I forgot to guess them through the hole in the top of the plug. I've got more plugs at home with that hole out through than I would care to mention. I don't think I'm alone either.

I connected the keyboard to the monitor, plugged in and switched on. And switched on again. The screen was a delight and the character set looked nice too. I curbed my natural desire to start typing and sat for a few minutes with the manual, just in case I had missed anything on my first glance. Considering that this had been in the pub a couple of days previously I thought better safe than sorry!

The first useful point that struck



ALAN McLACHLAN starts a regular column prowling the whacky whimsical world of the Amstrad CPC464

me was the apparent ability to type in lower case — variables and keywords alike. I tried it.

```
ll cin
ll print'hello'

not

hello
Ready
```

You see, I've read lots of beginners' articles and according to cockat you should always print "hello" at your first attempt on the keyboard. So far so good! Right, we'll fix it.

```
1.
Syntax error
```

First mistake. The BBC accepted abbreviated commands *ll*, was short for *LIST* but the Amstrad doesn't. I wasn't bothered about it but I must have made the same mistake at least 50 times during the first week. Old habits die hard. Try again, Al.

```
list
ll cin
ll PRINT'hello'
```

It worked. As predicted the command words, because they are "tokenised" — they are recognised by the computer and allocated a particular code — appeared as capitals.

This proved to be a very useful aspect of the machine because identifying incorrectly spelt commands became simple. And being fairly simple I typed a lot of incorrect commands! They remained in lower case, silent witnesses to my incompetence.

```
ll cin
ll print'hello'
ll print'hello'

not

hello
Syntax error on ll
ll print'hello'
```

What's happened is that the machine has executed the program up to the error and not only indicated the error line but has also listed it in edit mode. By this I mean that the line is now ready for editing using the cursor keys. I'll explain this later when I understand it myself.

The important thing to notice here is that because "print" was spelt wrongly, it did not reappear as *PRINT* in the listed line.

This is an extremely useful facility, particularly when entering long listings. Normally my programs have more bugs in them than a down and out hotel (and to be able to spot syntax errors quickly is a great de-bugging aid).

And while we're on the word *PRINT* I'll mention another interesting point. I'd typed it in numerous times before I discovered that there was an abbreviated form. I'd been trying to talk them out the colour commands you know:

```
The ink is black,
The paper's white,
Wrong way round and
It's out of sight.....
```

Well I've passed it out now (I think) and I'll tell you about the experiments another day. At the time though, I gave it up as a bad job and took the manual to bed. (There's no

way a manual can use a headline as an excuse for non-cooperation.)

I started at the beginning again and found the helpful hint about the ? replacing the *PRINT* command. I was tempted to keep out of bed and dash back to the machine, but managed to curb my enthusiasm when I glanced at the clock — 2am.

The ? is very handy and particularly useful in replacing those infernal P, syntax errors. And of course, it comes out as *PRINT* when the program is listed. (How does it know? *PRINT*!)

```
ll cin
ll ?'hello again'

list

ll cin
ll PRINT'hello again'
```

This abbreviation is not new to me as I came across it on the Dragon.

Another useful abbreviation on the Dragon which is also used on the Amstrad is the *REM* command. The computer ignores everything after a *REM* and this is extremely useful for making little notes within your programs to make them easier to follow.

```
ll cin
ll PRINT'for(jan)'REM French for
hello

ll PRINT'hello': ' short for REM
```

The abbreviation is on the shifted ? key [1]. It does save a few letters of typing, but is not quite as obvious as the full statement. Because of this, listings in Computing with the Amstrad will use the full *REM* rather than the abbreviation and you would be well advised to follow suit.

Well, I think we'd better call it a draw for this month. All this new-fangled technology is a bit heavy on the eyelids and I really should get my beauty sleep. Good bye, does he need it - full!





National Micro

AMSTRAD CPC464

The COMPLETE computer

Everything you need to start you computing immediately – includes a specially designed monitor and built-in data cassette recorder.



**PRICES
FROZEN!**

While present stocks last

£349
colour

£239
green screen

Carriage 17

Also available

Printile

Power

Fantale



All our prices
include VAT



The ideal printer for the CPC464

**Mannesmann
Tally MT80
dot matrix**

£217

Price includes lead
for interfacing printer
to the CPC464

Carriage 17

BOOKSHOP



A choice selection of 28 top rating games that'll give you hours of fun. It also shows you how to incorporate ready-made routines into your own programs.

£6.95



Make the most of your Amstrad's potential with this information-packed book. It's full of easy to follow information that's certain to improve your programming.

£6.95

Starting where the basic instructions leave off this invaluable book explores the Amstrad's sound, graphics and assembly language capabilities to the full.

£8.95



If you're baffled by the User's Instructions, this is the book for you. Its step-by-step no-nonsense introduction to Amstrad Basic is highly recommended.

£7.95

Put your Amstrad to good use with these forty games, full of educational value. Subjects include languages, geography, maths and science.

£8.95



If you've ever wondered what goes on behind the scenes in adventure games, or thought about writing one, this well thought-out book is the one for you.

£7.95

ORDER FORM

All prices include post and packing

Respective authors required to be

£

- | | |
|--|-------|
| <input type="checkbox"/> 48 Educational Games for the Amstrad by Mike Apps (Oxford) | £6.95 |
| <input type="checkbox"/> The Amstrad Program Book by Peter Smith (Penguin) | £6.95 |
| <input type="checkbox"/> Adventure Games for the Amstrad (EPIC) by A.J. Bratby (Collins) | £7.95 |
| <input type="checkbox"/> Amstrad Computing for the Beginner (Oxford) | £6.95 |
| <input type="checkbox"/> Sensational Games for the Amstrad by Jim Briggs (Oxford) | £6.95 |
| <input type="checkbox"/> The Amstrad CPC464 Explained (Routledge) | £8.95 |
| <input type="checkbox"/> The Amstrad CPC464 Advanced User Guide by Mark Harrison (Digital Press) | £7.95 |

I enclose my cheque/£ P O for £

Name

Address

Post to Database Publications, Eyre Road, Basingstoke, Hampshire RG24 0PT

Cheques made payable to Database Publications Ltd

These prices apply to UK readers only. Overseas prices on request.

WATCH THIS SPACE FOR NEW BOOKS EVERY MONTH, TO DEVELOP YOUR KNOWLEDGE OF THIS EXCITING MACHINE STILL FURTHER!



Give your fingers a rest!



ALL Suggesting **AMSTRAD**
programs
on one
cassette
for only
£3.75

Why tire your fingers typing when you can get all the programs from this issue in one value-packed cassette?

Take a look at what's on offer and you'll see what we mean. We've games galore, useful utilities - plus lots of exciting extras...

SMILEY: Can you avoid the Grumpies as you guide Smiley round the labyrinth? Our Amstrad version of this arcade favourite guarantees hours of fun.

CODE: You don't have to be a mastermind to play this intriguing logic game - but it helps!

BINARY: Baffled by binary bias? Let our utility help you out.

DANCER: Simple but fun, our dancer is a lovely little mover.

TRAPPER: You'll need quick-thinking, fast reactions and downright cunning if you're going to successfully pen the malevolent monster of the maze.

SCROLLER: Add that professional touch to your text with this slick sideways scrolling routine.

LETTER LITTER: Keep your Amstrad tidy and learn the keyboard layout at the same time with this entertaining educational game.

PLUS all 13 example programs from our Sound and Graphics articles.

*You've read
the mag -
now load
the tapes!*

HOW TO ORDER

Please send me the cassette of programs from the January issue of Computing with the Amstrad.

☐ I enclose cheque for £3.75
made payable to Commodore
Publications Ltd.

☐ I wish to pay by ☐ Access ☐ Visa

No. _____

Expiry date _____

Signed _____

Name _____

Address _____

POST TO: Amstrad Tape Offer, Europa House,
68 Chester Road, Hazel Grove, Stockport S87 5NY.

I THINK the question I get asked most often by micro enthusiasts is: "What exactly is machine code?" I just can't make sense of all this CD MAC, LS and JP 802 business. I bought a book, but that didn't help".

Well, this series of articles is an attempt to answer that question. You may not be an accomplished machine code programmer at the end of it, but you will certainly know what machine code is, and be able to write your own simple programs.

Better than that, you'll be in a position to take advantage of the many excellent books on 286 machine code currently on the market, and see how they fit in with your Amstrad. From then on you'll be able to teach yourself, and that's always the best way.

So what IS a machine code program?

Well, let me dodge the question by telling you that all programs are machine code, eventually, and we'll get round to exactly what that means in a minute.

First of all, tradition decrees that I tell you that the microprocessor at the heart of the Amstrad CPC486 is the 280A, complete with 8 and 16 bit registers and a 16 bit address bus. I should then go on to discuss its arithmetic logical unit, its internal data bus and so on, referring you to an incomprehensible diagram showing its "architecture".

To hack with all that. Let's talk about it from the consumer's point of view - yours. You see, I'm not one of those "you can drive a car better if you know what's under the bonnet" freaks. I have it on good authority that gynaecologists do not make the best lovers.

So what is machine code all about? The fact is, it's all about numbers - lots of them. More precisely, it's about lots of numbers, each of which is between 0 and 255 in value.

Show me a machine code program and I'll show you a load of such numbers. Forget about LD and JP for the moment. Believe me, it's all done by numbers.

Let me explain. We've used to talking about a micro having memory aren't we. Well a micro's memory is composed of lots of individual memory cells, as is our own brain.

It's all done by numbers

MIKE BIBBY helps make sense of machine code

And, just like our memory cells, a micro's memory cell can only remember so much.

In the case of the 286, the cell can remember only one byte at a time - and a byte, you won't be surprised to learn, happens to be a number in the range 0 to 255.

An upper limit of 255 might seem a little arbitrary, but there's an excellent reason for it. It's all to do with the wiring. (Okay, we'll do the bonus just a little!)

Each memory cell, or location as it's more properly termed, consists of a set of eight switches, each of which can be either ON or OFF. Now by arranging the switches in various patterns of on and off, we can encode things - a sort of electrical semaphore. And what we code is - yes, you've guessed it - numbers!

Have a look at Table 1. What it does is to link each switch with a number. We've labelled our switches switch 0, then switch 1 and so on up to switch 7. Notice that, yet again, computers start counting at 0. Even though we only go up to switch 7, there are eight switches in all.

Now below each switch in the table is the number linked to it (don't worry who we picked these particular numbers for the moment). Switch 0 is

worth 1, switch 1 is worth 2, switch 2 is worth 4 and so on.

Notice how the value of each switch doubles as you go along. Given these values we can code numbers. For example, if switch 4 were ON, and all the others off, we'd be "telling" the number 16. Similarly, if just switch 7 were on, we'd be telling 128.

Even better, by having more than one switch on at a time we can code other numbers than just the eight we've linked so far - we just add the values of all the switches that are ON, to arrive at the new number. For instance, if switch 5 and switch 1 were on simultaneously, and all the others were off, the number we've got is 34. Figure 1 shows why.

If you think about it for a moment, you'll see that the smallest number we can encode is 0 (all the switches OFF) and the largest number we can encode is 255 (all the switches on).

The really nice thing about the way we've chosen our numbers though, is that every number between 0 and 255 has its own unique pattern of switches, so there's never any confusion about the number you've coded, or stored - to use computerese - in the byte.

But, and it's a big but, writing 34 as

Switch	7	6	5	4	3	2	1	0
Value	128	64	32	16	8	4	2	1

Table 1: Values associated with each switch

Switch	7	6	5	4	3	2	1	0
Value	128	64	32	16	8	4	2	1
State	off	off	on	off	off	off	on	off

gives ---- > 32 + 2 = 34

Figure 1: Encoding 34 with switches

off off on off off off on off is incredibly cumbersome. However mathematicians decided that since there were only two states for each switch (ON and OFF), they'd use the number 1 for ON and 0 for OFF. Using 1 and 0 in this way gives us what are called binary numbers. In this scheme of things 108 becomes 1001101010. Figure 11 shows how.

You may be wondering why we've put the 16 in front of the 01101010. The reason is that otherwise we might mistake it for an incredibly large ordinary number. So if you see a 16 in front of a number it's coded in our binary way. Incidentally, each switch is known as a bit, and since a byte consists of eight such switches, we can say that there are eight bits in a byte. The article Bits and Bytes on Page 38 goes into it in more detail.

Now these eight bits in a memory byte allow us to store any number from 0 to 255 — 256 different numbers. If you remember to count 0. But if we're going to have a computer of any power we're going to need more than 256 bytes of memory.

So what the micro does is to have 65536 different memory locations, numbered from 0 to 65535, to store its data in. Why 65536? Well, in order to keep track of its memory bytes, the computer has to do some more wiring.

We've already seen that having eight wires would only allow us to keep bits on 256 locations. What the 280 does is to double up the number of wires to 16 — which then gives it 65536 as its largest number. Look at Table 1 if you don't believe me.

As you can see, it's like the old

16 byte	15	14	13	12	11	10	9	8
value	32768	16384	8192	4096	2048	1024	512	256
10 bit	7	6	5	4	3	2	1	0
byte value	128	64	32	16	8	4	2	1

Table 1: 16 bits explained — compare with Table 1

Switch	7	6	5	4	3	2	1	0
Value	128	64	32	16	8	4	2	1
State	off	on	on	off	on	off	on	off
Binary	0	1	1	0	1	0	1	0
Giving ———> 64 + 32 + 8 + 2 = 108					decimal binary			
————> 1001101010								

Figure 11: The binary representation of 108

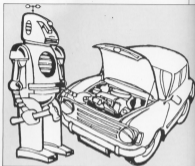


Table 1 with an extra eight switches or bits added on top — that is, another byte.

To get the value of these extra bits we just keep on doubling. 128 was the last one, so it goes 256, 512 and so on up to 32768. The top (higher valued) set of eight bits is called the high byte of the address — hi byte for short. The bottom (lower valued) set of eight is called the low byte of the address — lo byte for short.

If all the switches are on — that is, if

all the bits were set at 1 — the number these two bytes would code is:

16384	16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1
65536															

Now do you believe me? The reason we've gone into as much detail is because, as I've said, machine code is all about numbers stored in the micro's memory. In fact machine code is mostly about moving these numbers (and hence the information encoded in them) around the memory of the computer — that is,

from one memory location to another.

For example, there's a large machine code program that actually runs your CPC464. It's called the operating system, or firmware. One of its jobs is to print that familiar "welcome" message on the screen when you first turn on the machine.

What happens is that the message is stored away in the micro's memory. When you switch on it copies the message from those locations into the memory reserved for the screen so you can see it. That is, the firmware machine code program receives the numbers that encode the message from one location to another.

This same sort of transfer of data occurs when you press a key. The firmware transfers the value of the key pressed from the location that remembers which key it was to the memory set aside for the screen.

When you save a Basic program the firmware's own machine code program moves the contents of the memory where the Basic program is stored out to the cassette port.

It's all about moving bytes of data around! More formally, most of machine code is concerned with moving bytes of information from one memory location to another. If you're a realist, you'll probably have guessed that there's a lot more to it than that. But have faith, most of what I'm telling you is true.

To investigate this movement of bytes further we need an analogy – in other words a meaningful lie. Suppose we have a tiny micro with only three memory locations. Figure III shows the sort of thing.

It's fairly easy to wire them up so that the numbers can move from one location to another – just join each byte to every other byte in the figure.



Figure III: Linking three memory locations

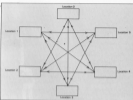


Figure IV: The complexities of linking six memory locations



Figure V: Memory locations linked by A register

by the way, I've only shown one of the eight wires for clarity).

If you stretch your imagination you'll see that it looks a lot like a simple railway.

But suppose there were more locations, as in Figure IV. You can see the layout's getting complicated. And when you consider that the 280 addresses tens of thousands of such locations, you can see that we've got problems – the wiring's far too complex.

Of course the answer is to stop giving each memory location its own direct lines to each and every other location. We'll do what railways do, and have junctions and branch lines.

Figure V shows such a layout. Our six locations are all connected via the major junction A. Everything passes through here. In fact there is such a memory location as A – a major junction through which traffic passes. It's deep in the heart of the 280 and can hold one byte numbers. In computing jargon we call such a junction a register.

Now suppose you wanted to move

a byte of information from memory location 0 to memory location 5. As you can see from the figure, you would go via register – that is, junction – A.

You would do this by giving the machine two instructions:

1. Load register A with the number contained in memory location zero.
2. Load memory location 5 with the number that is in register A.

It's a set of microelectronic pass the parcel. The data goes from location 0 to A, then from A to 5. It's always a two-stage journey. All traffic passes through A.

In practice the actual layout is more like Figure VI, but there's still no direct traffic. Everything goes to A first and then back out.

To stretch our analogy a little further, a major rail junction like A would have lots of facilities that other junctions haven't. It's the same with the 280 – since you've got a number in A you can do all sorts of things with it!

Also no rail designer worth his salt would depend on one major junction

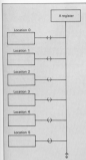


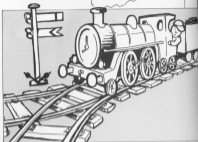
Figure VI: A more realistic representation of memory/register linkage

— there'd be too much congestion. We'd have other junctions. Similarly the Z80 has registers other than A for much the same reasons.

And as it stands, our junction/register A doesn't have all that much capacity — just one eight-bit number. This could be limiting if we wanted to deal with larger numbers, such as we use to label memory locations. Well, the Z80 has got registers to handle these, too, as we'll see later on in the series.

By now I think I've convinced you that machine code's all about moving numbers around in memory. But how does the micro know what to do? How do you tell it to move these numbers, and where you want them going?

The answer's simple — you give it a



list of numbers stored in memory!

I'm not joking, honest.

The program itself is just a sequence of bytes in memory. The bytes have meaning to the Z80, you see — it's a sort of code, machine code in fact. All you do is point your Z80 at the first byte and say go. It then moves along the list of bytes doing what it's told.

Let's have a look at what this means in the context of the little program we discussed above — the one that transferred one byte from location 0 to location 5.

The actual string of bytes we need is:

58 00 50 50 50 301

I've written the numbers in decimal, as that's what we're used to — of course the micro reads them in binary. Figure VII explains what it's all about. When we point the Z80 at the location of the first byte of our program and tell it to go it knows that first byte is going to tell it to do something. The fancy name for this sort of "command" byte is an opcode — short for operation code.

Now 58 is an opcode that tells the Z80 to load register A with the contents of a particular location in memory. From the opcode itself, the Z80 knows that the address of this memory location will be contained in the two bytes directly following the opcode (remember you need two bytes to specify addresses).

So having understood the meaning of the opcode, the Z80 moves its attention along to these two bytes and works out the address they refer to. In this case, location 0. It then copies the contents of that address into the A register.

The Z80 has finished with the first three bytes and has done all the first opcode instructed it to. It now turns its attention to the fourth byte, which it knows must be an opcode since it has finished its previous task.

This time the byte is 50, which tells the micro to load the memory location specified in the next two bytes with the number in the A register. On a sense, this is the mirror image of the last opcode. That loaded the accumulator from a memory

Contents of memory location	58	00	00	50	50	00	301
Meanings of above bytes	Load A with contents of the address that follows	These two bytes specify the address needed by previous opcode		Load the address following with the contents of A register	These two bytes specify the address referred to by the previous opcode		Return from whence you came

Figure VII: A simple machine code program explained

Machine Code

location. This opcode loads a memory location from A.)

So having worked out what the opcode contained in the fourth byte wants it to do, the Z80 turns its attention to the next two bytes along, works out the address, then does all copies the A register into that location.

Having finished that instruction, which used the fourth, fifth and sixth bytes, the Z80 then moves on to the seventh and last byte to find its next opcode.

The seventh byte contains 201, the opcode for return—which tells the Z80 to go back to where it was before it started in, as the jargon has it, called this program.

This works in much the same way as RETURN does in a subroutine, causing the micro to rejoin the main flow of the program.

Notice that you don't need any extra bytes after this opcode to tell it where to return to. When this routine was called the Z80 carefully stored where it was up to for future

reference—as does a Basic program when it meets a GOSUB.

By the way, you may have noticed that the two bytes specifying location five are not 0,5 as you might expect, but 5,0.

I don't want to go into this too much this month. Suffice it to say that the Z80 likes to know the 16 bytes of an address before it receives the hi byte.

Let's have another look at the machine code program we've developed. I'm going to split each instruction—that is each opcode and its data bytes—onto a separate line.

```
55 0 0
55 5 0
201
```

Doesn't make immediate sense, does it? Our brain is much more adept at making sense of words than numbers. Have a look at the program in a new form, that uses "words":

```
LD A,00 55 0 0
LD 5LA, 55 5 0
RET      201
```

The symbols on the lefthand side

are mnemonics. LD stands for Load and RET for RETURN.

The translation is as follows:
LD A,01 Load the A register with the contents of memory location 0.

LD 5LA Load memory location 5 with the contents of register A.

RET Return to the program that called the machine code in the first place.

You can get special programs called assemblers that let you type in your routines in these more meaningful mnemonics and then translate them into machine code, but they're a luxury we'll be doing without for a while.

Well that's all for now. I hope you've got a better understanding of what machine code is.

■ Next month we'll be looking at hexadecimal and running your own machine code programs. Until then, take a look at *Bits and Bytes* on Page 35. And practice your binary—you'll be needing it!

• NOW AVAILABLE FOR THE AMSTRAD CPC 464! WIN THE POOLS?

SPECTADRAW 3 - THE LATEST VERSION OF THE ORIGINAL AND BEST POOLS PREDICTION PROGRAM FOR THE AMSTRAD CPC 464!

AND NOW... AMSTRAD ORANGE - THE FIRST POOLS PREDICTION PROGRAM FOR THE A MACHINE WITH AMSTRAD CPC 464!



- Supplied with Database containing data on over 10,000 matches since 1890!
- You update the Database each week - fast & without typing, as team and division names already in program!
- Scores easily converted - the program even checks your entries!
- Comprehensive instruction manual and menu driven program - easy to use, even for a newcomer to computing!
- Will forecast the best team to win for those who prefer to bet on fixed odds!
- Built in prize generator - complete your coupon direct from the screen!
- Fully microdrive compatible! (Spectadraw only).
- Compatible with Camm Micrograph - the first pools program to read its predictions! (Spectadraw only).

Spectadraw 3 for the 464 Spectadraw £9.95 inclusive
Amstrad Orange for the Amstrad CPC 464 £9.95 inclusive
(Cheques/P.O.s payable to B.S. MARLEY)

We dispatch every Monday with the database made up to include all matches up to the date of dispatch.

SPECTADRAW Dept A11
1 Caversham, Oxford, Oxford OX4 4TD.
(Tel: 0844 524266)

MINDBENDING GAMES for the AMSTRAD CPC



Philip Laird

An Ideal Gift for Amstrad enthusiasts
Hours of brain-teasing enjoyment for
all the family!

By Philip Laird. Published by another Publisher
£2.95 from bookshops or £3.95 by post
from the sole distributors

ANDREW BISHARA

The Glass House, 5-13 Blenheim Street, Ipswich, Suffolk IP1 1LP

Going for a scroll

SCROLLER is a short, simple program that scrolls a message letter by letter from right to left across the screen, repeating it endlessly in a "wrap around" display.

When you run the program it

asks you for the message you want displayed and the line you'd like it to scroll across. The program does the rest, displaying the message until you press a key.

If you want to incorporate Scroller in your own programs it

couldn't be easier. The subroutine that does the work (GOSUB 270) can just be remembered and merged with your programs. All you have to do is to set up the two variables mentioned in its REM statements. **Pete Bilby**

SUBROUTINES

GOSUB 30: Checks and accepts the message to be displayed, adding a space to it. It also accepts the line the scrolling is to take place on, again trapping erroneous input.

GOSUB 270: Does the actual work using two FOR...NEXT loops. The first loop has the message appearing from the right, growing letter by letter, travelling towards the left. This loop stops when the whole message has completed its journey from right to left and is displayed on the screen.

The next loop carries on the scrolling, displaying the message in the centre of the line, giving a wrap around effect. This is achieved by taking a letter off the front of the message, adding it to the end of what's left and then overprinting the original with the new string. The delay loops of lines 400 and 460 can be left

out if you want to see how fast your micro can work.

VARIABLES

Hold the string to be scrolled plus a space which separates the messages.

Determine the screen line that the display is to use.

The length of the message.

At first this is full of spaces, but each time the first loop cycles a space is removed and a letter added in its place until it eventually holds the whole message. The second loop uses it to contain and display the dismembered message as it wraps around.

message\$,
submessage\$

aline, subaline

sublength
subdisplay\$

substartposition

Holds the position of the left end of the display, which is fixed in line 340 so that it's centred on the line.

```
10 REM SCROLLER
20 REM PETE BILBY
30 REM (c) DATABASE PUBLICATIONS
40 MODE 1
50 GOSUB 30:REM Accept message
60 GOSUB 270:REM Scroll message
70 END
80 REM*****
90 REM Accept and check message
100 LOCATE 1,10
110 PRINT "What message do you want a
  scrolled?"
120 LOCATE 1,12
130 INPUT message$
140 IF LEN(message$) < 50 THEN 100
150 REM put a space at the end of the
  message
160 message$=message$+" "
170 LOCATE 1,10
180 PRINT "Which line do you want it
  to appear on?"
190 LOCATE 1,12
200 INPUT aline
210 IF aline < 10 THEN 200
220 submessage$=message$
```

```
230 subaline=aline
240 CLS
250 REM
260 REM*****
270 REM Scroll the message
280 REM the subroutine needs three va
  riables when called
290 REM subline holds the line positio
  n
300 REM submessage$ holds the string
310 REM the delay loops slow things a
  bit
320 sublength=LEN(submessage$)
330 subdisplay$=STR$(sublength) * "
  "
340 substartposition=INT((100-sublength
  )/2+1)
350 REM first loop has string appeari
  ng from right
360 FOR loop1 TO sublength
370 subdisplay$=LEFT$(subdisplay$,lo
  op1)+MID$(submessage$,loop1,1)
380 LOCATE substartposition, subline
390 PRINT subdisplay$
400 FOR delay1 TO 500:NEXT delay
```

```
410 NEXT loop
420 REM second loop just cycles the a
  right message
430 REM until a key is pressed
440 WHILE INKEY=""
450 subdisplay$=MID$(subdisplay$,lo
  oplength-1)+LEFT$(subdisplay$,1)
460 LOCATE substartposition, subline
470 PRINT subdisplay$
480 FOR delay2 TO 500:NEXT delay
490 REM
500 REM
510 REM*****
```



Give your fingers a rest...

All the listings from this month's issue are available on diskette. See our special offer on Page 22.

Starting at ground zero

A MISTRAD Computing with the CPC464 is the latest offering from the prolific pen of Ian Sinclair. Aimed at the newcomer to the Amstrad, the beginner to programming of any sort is certainly not forgotten.

Mr Sinclair says in his opening paragraph: "The Amstrad CPC464 computer offers much more for either the home or the business user than any previous machine at a comparable price, more even than many machines at much higher prices. Because of this many buyers of the machine will never have made use of a computer before, certainly not a computer with the range of facilities that the CPC464 offers".

He certainly seems impressed with it, but is not prepared to ignore trivialities such as finding the key tops loose when he opened the hardware package for the first time. He made it sound the most natural thing in the world and included an explanation of how to put the speaker back on with the aid of a pair of hammers.

The book starts off like all good quality beginner's books should - at "ground zero". The first chapter covers first and foremost, in detail, how to attach a five amp plug. Why not? The manufacturers couldn't be bothered to fit one, so let's get things right from the start! Connecting to a monitor/TV, and tuning the signal is next on the agenda.

This is followed by a short introduction to some of the command keys such as F10, Ctrl, Enter and Ctrl and a comprehensive discussion of loading and saving.

The book then takes you gently through the PRINT command using numbers and strings and an explanation of the functions TAB and LOCATE. In no time at all you are using string and number variables

with INPUT, and even defining a simple function in the following manner:

```
1 DEF FN SQR(X,Y)=X*Y
IF GJ
20 PRINT "Enter three numbers, please"
30 INPUT A,B,C
40 PRINT "You la " ; FN SQR(A,B,C)
```

Nothing earth-shattering here I must admit, but the way it is explained makes it natural to find such "advanced" techniques so early in the book.

There's an excellent chapter on loops which starts off with the dreaded GOTO and proceeds through FOR...NEXT to the completely new - to me - WHILE...WEND.

As Mr Sinclair states: "Very few home computers offer you any more than a simple FOR...NEXT loop...the CPC464 however offers you a very different kind of loop. This can often make it much easier to program".

I have got to find this out. I trust him implicitly and will persevere. Mind you, it seems so much easier to fall back on a FOR...NEXT when things get tricky.

There are sizeable chapters on menus, subroutines and file handling, and all are explained in an interesting, easy-to-read manner. I have still fully to get to grips with file handling but this book certainly goes a long way to making it a little clearer for me.

The graphics section covers three chapters and gives a comprehensive account of the WINDOW, BORDER, PAPER, PEN and INC commands.

In chapter 10, "Guide to greater graphics", the MOVE, PLOT and DRAW commands are explained with excellent examples. However nowhere could I find the answer to the one question that has been bugging



me since I started programming the CPC464: "How do you set a foreground colour while PRINTing at the graphics cursor (TAB), without using the PLOT command with its colour parameter?"

I can't find any other solution than plotting a point off screen. I'm sure I'll be enlightened soon by someone who has found it oh so simple.

The CPC464 has vast SOUND potential with its two envelopes for both sound and tone, and Mr Sinclair gives quite a successful detailed explanation of both.

It was as easy to understand as such a computer subject can ever be if it can be, and I feel more confident in tackling these facilities having read this exposition.

To cap it all the book rounds off with appendices on printing, programming keys to do special things and error trapping.

The many useful and well annotated listings were all taken straight from the Amstrad and therefore should contain no errors. Having heard this claim before I was somewhat sceptical, but must admit that all the examples I tried worked perfectly.

All in all this book would make a useful supplement to the Amstrad's own manual and at £6.95 is good value for money.

Alan Sergeant

Journal Computing with the
CPC464 (Canda, £6.95)

*Now you've started computing with the Amstrad
you'll want to read EVERY issue of...*

Computing *with the* AMSTRAD

It's the only way to keep right up to date with what's going on in the ever-expanding world of the Amstrad. Look what you will get each month:

SPECIAL OFFERS!

How to protect your CPC464

Protect your micro with our luxury dust cover made of soft, pliable, clear and water-resistant vinyl, bound with strong cotton and decorated with the magazine's logo.

DUST COVER ONLY \$1.95

How to keep your collection complete

Bound in rich burgandy type and bearing the Computing with the Amstrad logo, this handsome binder will hold a year's supply of the magazines, firmly secured in place with metal rods.

FINCH ONLY \$3.95

- ★ Reviews of all the very latest games, educational and business programs now being produced for the Amstrad.
- ★ Lots of listings you will be able to key in yourself – games, utilities, graphics – covering the whole field of Amstrad computing.
- ★ In-depth independent evaluations of all the new hardware add-ons now being developed to make your Amstrad much more powerful and much more versatile.
- ★ Lots of easy-to-follow features on everything to do with the Amstrad. Whether you're a beginner or an expert, you'll always find something to delight and intrigue you.

We're going to make this the most exciting computer magazine ever - so don't miss an issue! If you've got a CPC464, or about to get one, let Computing with the Amstrad show you how to make the most of it!

INTRODUCTORY OFFER!

Take out a 12-month subscription now and save only \$19 instead of the normal \$12

This special offer applies to all 16 members only and is valid until January 31, 1986.

SAVE \$21

SUBSCRIPTION FORM

1. **Introduction**

© 2000 by John Wiley & Sons, Inc.

© 2000 by The McGraw-Hill Companies
All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or by any information storage and retrieval system, without prior written permission from The McGraw-Hill Companies, Inc.

100

(continued)

(Faint handwritten notes)

Send no Cheques/Payments
To: 70024021, Service Team,
4th Century Books, David Cooper,
Newquay PL4 8DZ

THE UNIVERSITY OF CHICAGO

Attention all Amstrad owners!

SOFTWARE SUPER SAVERS

KARLS TREASURE HUNT



KARLS TREASURE HUNT

Our hero Karl has fallen on hard times so when he learns that his entry has won 1st prize in a quiz competition he is naturally delighted. His prize is a weekend stay at 'Wonga Mansion' and hidden somewhere in the mansions 40 rooms is a treasure chest. Karl has to collect the 40 keys to unlock the chest. Can Karl's luck be changing for the better.

£2.99

Available on the Amstrad CPC 464

Software Super Savers is a new name to watch out for. We'll be bringing you quality software at a super-saver price. They're not ex-husbands of old games but totally original ideas combining to give you an exciting range of new games.

So whatever your software tastes are, Software Super Savers has the game just right for you.



Dealer Enquiries -
051-428-0307 and
ask for Lesley

Software Super Savers Ltd.,
P.O. Box 13, Liverpool L26 3AG

Please send me a copy of

KARLS TREASURE HUNT

I enclose cheque/PO for

(Please add £1.00 for orders outside UK)



Access Card No.

Name

Address

not that letter only

Software Super Savers Ltd., P.O. Box 13, Liverpool L26 3AG

I **THOUGHT** you would be interested in a little hint for keeping track of sub-routines without searching through the whole program.

The idea is to have all the sub-routines' first line numbers listed at the start of the program. The list consists of GOTO statements to point to the routines. For example the program may look like this:

```
10 GOTO 100
20 GOSUB 3000: REM
   Create characters.
30 GOSUB 3000: REM
   Initialise screen.
40 GOSUB 4000: REM
   Instructions.
50 GOSUB 5000: REM
   Move invaders.
60 GOSUB 6000: REM
   Move ball.
70 GOSUB 7000: REM
   Fire laser.
80 GOSUB 8000: REM
   Highscore table.
90 GOSUB 9000: REM
   Game over.
100 REM *****START OF
   PROGRAM*****
2000 REM Characters.
3000 REM Screen.
```

When the program is reassembled the list of GOTOs is updated. The GOTO at the start of the program jumps round this list and therefore the GOTOs are never out of date.

I hope this will be of some use to your readers. — **Terry Blunt, Berkshire.**

■ This is a very simple tip but very, very useful. We will certainly be making use of it while developing our own programs — we are sure many of our readers will too.

Let's keep track of those sub-routines



Finished disc drives... on their way west

Discs and modems

I **WON'T** get an Amstrad yet, but I am seriously considering buying one as it seems to be a very competitively priced package.

I was wondering whether you could clear up a couple of points that would help me make up my mind.

I don't want to buy it if there is any risk that the disc system may never appear. Have you any comments on this?

Secondly, in this, the year of communications, I hope soon to purchase a modem. Will the Amstrad support one?

These questions may seem

trivial, but I don't want to go and lose out over £200 and then find out that it wasn't what I had hoped. — **David Hughes, Shrewsbury.**

■ We have it from a reliable source that both disc drives and communications hardware will be launched shortly.

Future of Forth

I **HAVE** spent nearly 12 months getting the hang of Forth on a Jupiter Ace. Also it is no more, and has gone to that great chip shop in the sky.

I understood the Amstrad will support high-level languages and I am thinking of buying one. Is there likely to be a Forth for it, and if so will it be in ROM or an cassette? — **Julian Bayliss, Weyford.**

■ We have spoken to Amstrad and they say Forth is not in their immediate plans, but it could well be on the cards for the near future.

It is a reasonably safe bet that if Amstrad don't do it someone else will.

Buffer bother

I **LEARN** to program on a BBC Micro at school but now I've got myself an Amstrad and

I'm relearning.

The problem I've got is how to empty the keyboard buffer. On the Beeb this is easy but I can't find any way of doing it on the CPC.

As it is, I keep getting stuff like PC700 at the keyboard that I don't want. Any ideas? — **Ian Worrie, Glasgow.**

■ To be honest, no. There must be a simple way of doing it but all we can think of is to wait a line such as:

WILE INPUT="" : END

just before you go to the keyboard to look for an input. This clears out any garbage and leaves the micro ready to receive fresh input.

Anybody got any better ways?

Sound advice

ONE of your writers has informed me that you're doing a magazine for the Amstrad CPC464. If so, this could be your first letter.

Can you answer a question for me? I've been messing around with the SCL160 system but things keep going out of step.

The modules I make for, rather, the micro makes are out of sync with the program. I've only been using channel 0. A try rate as on there is

Computing with the AMSTRAD Postbag

WE welcome letters from readers — about your experiences using the CPC464, about tips you would like to pass on to other users... and about what you would like to see in future issues.

The address to write to is:

Postbag Editor
Computing with the Amstrad
Ramage House
68 Chester Road
Hazel Grove
Stockport SK7 5NY

carefully B? — **Tom Price, Wolverhampton.**

■ Surprisingly you're not our first letter, but time seems to have gone before us! As for your problem, it sounds as though what's happened is that you've discovered the sound square.

As you know, Amstrad Basic is very fast and it can whip through a short program in a fraction of a second. The trouble is that we usually want the sounds we make to last for a second or two, but we don't want the program to stop while they're playing.

The CPC keeps record of this by popping the notes on a sound queue which processes them independently while it gets on with the Basic.

Your problem occurs when you want the program to coincide with a sound at a certain point in its execution.

You can do this using the SO function which tests the state of a channel. A line like:

```
WHILE SOUND= 127 :GOTO
```

holds up the program by cycling round until channel A is free. You can use this as a means of waiting the end of a note coincide with the next bit of Basic.

Epson link-up

I HAVE connected an Epson FX80 to my Amstrad CPC464 with the help of a knowledgeable person, and have encountered a few problems.

Each time I hit my program I get double line feeds. By this I mean a blank line between each program line.

Another problem is that character codes sent to the printer greater than 127 are misinterpreted.

Have you any ideas as to why the Amstrad is rejecting the codes? — **Paul Fisher, Bristol.**

■ The problem with line feeds can be solved by cutting wire 14 of your printer ribbon cable. Actually finding the fourteen can be a little tricky. Starting at the side with the raised wires, there's one 1, 19, 3, 23, 3, 21 and so on. If you've counted

correctly the wire 14 should be the eighth in from the un-raised side.

As you correctly state, all code greater than 127 produce unusual effects. This is because bit 7 of each byte sent to the printer is ignored. This has the same effect as sending the character ANDed with 127.

What makes it worse is that both the hardware and software allow only 7 bits.

If any reader knows how to get round this problem, please write in with the solution. By the way, there's a screen dump for the FX80 coming up in a future issue of Computing with the Amstrad.

Random reflections

AS the owner of a BBC I decided recently to invest in the Amstrad as a second computer.

I was experimenting with

FOR had a BBC for about one year and have just bought a CPC464. I have found from a reliable source that the Amstrad has got a 6845 Cathode Ray Tube Controller (CRTC) — also found in the BBC computer.

I know how to program this chip directly on a BBC but I cannot find any way of doing it on my Amstrad. Do you know how this can be done? — **Frank Henderson, Manchester.**

■ From Basic you can access the 6845 chip by using the

the RND(X) function and noticed that whatever number I entered in the bracket the video responded with a six figure decimal number.

If this is correct why does the manual quote RND(5) as an example when RND(Any-thing) would achieve the same result?

It would appear that the only way to get random whole numbers is to use the INO function as a random number that has been multiplied by 10 or 100 depending on the range required. — **Matthew Gould, Winchester.**

■ The RND function only generates a number between 0 and 0.999999999, and this is the case whatever you include in the brackets.

The ideal way to select your problem is to use the expression $X = INT(\text{number} * RND(1)) + 1$ where "number" is the highest number in the range you require.

This will generate random numbers between 1 and

"number". If you need the range to start at 0 the expression must be changed to $X = INT(\text{number} + 1) * RND(1)$.

Colour quandary

COULD you help me? I'm absolutely desperate.

When I'm developing my own programs I often change the values of the pen/ink combinations.

When the inevitable errors occur, and I break into my programs to fix them, the listing often appears in quite bizarre colours.

What I want is a way to get back to the colour set that's there at start-up.

I know I can do this with Col = 255 + Esc, but I lose my program.

Do you know how to do it? — **Nanette Brown, Solihull.**

■ What you need to do is call SBC00 and all will be well. You can set up the small Enter key to do this for you like:

```
KEY 128, "CALL SBC00" +  
CHR(10)
```

We need to link it up with pen as follows:

```
KEY 128, "CALL SBC00 : PEN  
1" + CHR(10)
```

We also put a call to SBC00 at the end of our programs — it's only common politeness to leave things tidy!

ACCESSING THE 6845 CHIP

OUT command.

The 6845 Command Register is programmed by OUT &R000,X, where X is the register you wish to program.

The Parameter Register is programmed in a similar way except that the OUT &R000,Y command is used — Y is the number to be written to the previously selected register.

The 6845 is a very powerful chip. Games which use sideways scrolling effects rely on the 6845 to speedily recalculate.

We hope to cover the

subject of the 6845 in a future issue of Computing with the Amstrad.

Below is a simple program which shows how the 6845 can be programmed to change the start location of the video RAM — this is done by writing to registers 12 and 13 of the 6845.

```
10 OUT &R000,12  
20 OUT &R000,13  
30 OUT &R000,13  
40 OUT &R000,13
```

"You really can't go wrong with any Level 9 game
as they are all brilliant." *Crash Micro Sept 84*

RETURN TO EDEN

Level 9's first amazing full-colour graphical adventure.

Return to Eden is the long-awaited sequel to Level 9's top-selling Snowball adventure, set on the weirdest planet ever. Now it's here with 240 locations, masses of puns and puzzles and with hundreds of pictures in the AMSTRAD, CBM 64 and Spectrum versions.

"Whatever machine you own, if you have the slightest tendency towards adventure playing then you must try one of these games; unfortunately you'll probably end up wanting to buy the lot!"

— *Computing Today Aug 84*

"The Level 9 Adventures are superbly designed and programmed, the content first rate. The implementation of Colossal Cave Adventure is a nothing short of brilliant, rush out and buy it. While you're at it, buy their others, too, simply smashing!"

— *Play 24 June 84*

"Level 9 — arguably the producers of the best adventure games in the UK — have done it again. LOOK 4 Eden is a sparkling addition to its stable of winners."

— *Home User July 84*



Available from the WHLY Shop and good computer stores everywhere. If your local dealer doesn't stock Level 9 adventures yet, use the coupon to buy them from us, or ask him to contact: Distributor: Microdealer UK, Lightning, Lonsdale, BSL, Linn Tree, PCL (UK), MCC, Wonderlogic etc.

"One of the best adventure games I have ever had the pleasure to get my hands on. I can recommend Burrough Adventure without the slightest fear of being contradicted. This is a massive sojourn into the unknown."

— *Microlog Oct 84*

"The saga of time the whimsy... a remarkable adventure game. It carries all the hallmarks of a Level 9 Adventure — problems, text display and use of map — with graphics of a standard I have not yet seen before in an Adventure!"

— *Computer & Micro Games Oct 84*

"I thoroughly recommend these Adventure titles. They are excellent value for money, a good inspection of Adventure addicts should be without them. I believe Level 9 are producing a series of Adventures which should be regarded as classics."

— *Play 24 July 84*

Colossal Adventure: The classic computer game with a twist.

Adventure Quest: Magic power down through Middle Earth.

Colossal Adventure: A classic in the UK and USA — priced at £10.

Adventure: A classic computer game with a twist.

Adventure: A classic computer game with a twist.

Adventure: A classic computer game with a twist.

Adventure: A classic computer game with a twist.

Adventure: A classic computer game with a twist.

Level 9

Level 9

Level 9

Level 9

Level 9

Level 9

Level 9

Level 9

Level 9

Level 9

Level 9

Level 9

Level 9

Level 9

Level 9

Level 9

Level 9

Level 9

I ENCLOSE A CHARGE-PAY FOR MY ORDER
CASSETTE OR £11.95 PER COPY

My name:

My address:

My macro is

some of those listed below with at least

1000 Level 9 copies to:

LEVEL 9 COMPUTING

Care One, 129 Huggins Road
High Wycombe, Bucks, HP12 3PG



the only choice

Kuma

AMSTRAD CPC464

software



Holdfast



Game of Strands



Star Avengers



Galaxis



Music Composer



Logo



Database



ZEN Assembler



EASI/VAT



Home Budget

An outstanding selection from Kuma's rapidly expanding range of Entertainment and Application Software for the Amstrad CPC464 Micro-computer.

Book:

● **The Amstrad CPC 464 Explored**

This superb book is designed to let every CPC464 user, at whatever level, get the most from his computer. After an introductory section on the special Basic features, the book looks in depth at the excellent sound and graphic facilities including:

- Animation
- Windows
- Character sets
- Multitasking
- 3 Voice Tones
- MC routines for Basic
- Use of Zen
- Use of OS
- Sample programs

Available from your nearest Amstrad CPC464 Stockist.

Kuma Computers Ltd., 12 Horsetown Park,
Horsetown Road, Pangbourne, Berkshire RG8 7JW.

Please send full catalogue on Amstrad CPC464 products.

Name

Address

Phone

Trade Enquiries Please 07557-4305